Admissibility Algebra for Mathematical AI: Policy-Indexed Equivalence, Certified Quotients, and Proof-Grounded Transformation

Douglas C. Youvan

doug@youvan.com www.youvan.ai

July 7, 2026

Mathematical artificial intelligence will not be secured merely by larger models, stronger benchmarks, or more fluent theorem suggestions. Its central problem is admissibility: when may one mathematical object, proof, statement, model, or representation be substituted for another without destroying the identity that matters in context? Classical algebra studies lawful operations; abstract algebra studies structure-preserving maps; category theory and homotopy type theory refine equivalence beyond literal equality. Yet AI systems need an additional layer: an explicit policy that determines which equivalences license transformation, quotienting, memory, retrieval, proof search, and explanation. This paper proposes Admissibility Algebra as a mathematical foundation for proof-grounded AI. Its primitive judgment is $\Gamma \vdash X \equiv_\pi Y : C$, asserting that, in context Γ, objects X and Y count as the same under policy π with certificate C. Policies encode permitted invariants, certificates, obstructions, and scoped substitution rules. The resulting architecture treats hallucination as unauthorized equivalence, abstraction as controlled quotient formation, and trustworthy mathematical AI as certified transformation. The immediate goal is not general intelligence, but a formal corpus where sameness is indexed, checked, and machine-learnable.

Keywords: Admissibility Algebra, mathematical AI, policy-indexed equivalence, certified transformation, proof-grounded AI, formal verification, quotient formation, scoped substitution, equivalence relation, isomorphism, categorical equivalence, homotopy type theory, univalence, proof assistant, Lean, Coq, Agda, Isabelle, mathematical objects, invariants, certificates, obstructions, theorem proving, symbolic AI, neural-symbolic AI, hallucination, retrieval augmented theorem proving, admissibility policy.

A collaboration with OpenAI GPT-5.5 Thinking. CC4.0. 156 pages and 120 references.

https://orcid.org/0009-0004-4206-4253

Outline

1. The Problem: Mathematical AI Without Admissibility

1.1 Fluent theorem generation is not proof-grounded intelligence
1.2 Why plausibility is weaker than admissibility
1.3 Hallucination as unauthorized mathematical substitution
1.4 The hidden identity problem inside AI reasoning
1.5 From answer generation to certified transformation

2. Algebra, Abstract Algebra, and the Question of Sameness

2.1 Algebra as lawful operation
2.2 Abstract algebra as lawful structure
2.3 Morphisms as structure-preserving transformations
2.4 Equivalence beyond literal equality
2.5 Why mathematical AI needs the next layer

3. The Primitive Judgment of Admissibility Algebra

3.1 The judgment $\Gamma \vdash X \equiv_\pi Y : C$
3.2 Context Γ as mathematical jurisdiction
3.3 Objects X and Y as presentations, structures, or claims
3.4 Policy π as the rule of permitted sameness
3.5 Certificate C as the witness of admissible identity

4. Policies as First-Class Mathematical Objects

4.1 Why equivalence is not self-authorizing
4.2 Equality, isomorphism, equivalence, and homotopy equivalence as policies
4.3 Policy strength, weakness, refinement, and failure
4.4 The policy lattice of mathematical identity
4.5 Same objects, different policies, different judgments

5. Invariants: What Must Not Change

5.1 Invariant preservation as the core of admissibility
5.2 Algebraic invariants: order, rank, kernel, spectrum, dimension
5.3 Categorical invariants: universal properties, adjunctions, equivalences
5.4 Homotopical invariants: paths, higher paths, deformation classes
5.5 AI failure as invariant loss

6. Certificates and Obstructions

6.1 A certificate is not an explanation but a witness
6.2 Proof objects, maps, equivalences, homotopies, and computations
6.3 Obstructions as certified non-admissibility
6.4 Counterexamples, invariant mismatches, and failed transports
6.5 The admissibility record: admissible, inadmissible, undecided

7. Scoped Substitution

7.1 The failure of global replacement rules
7.2 Substitution only under policy
7.3 Transport across equivalence and its limits
7.4 Context-sensitive rewriting in theorem proving
7.5 The rule: equivalence without admissibility does not license replacement

8. Quotients as Controlled Collapse

8.1 Quotienting as permitted forgetting
8.2 The formula Ω/π
8.3 Useful abstraction versus destructive collapse
8.4 Quotient objects, setoids, congruences, stacks, and homotopy quotients
8.5 Mathematical AI as a quotient-management system

9. Proof Assistants as Admissibility Engines

9.1 Lean, Coq, Agda, Isabelle, and proof-object discipline
9.2 Formal libraries as admissibility corpora
9.3 The theorem prover as certificate checker
9.4 Tactic scripts as transformation traces
9.5 What proof assistants already know, and what they do not yet index

10. The Admissibility Database for Mathematical AI

10.1 Objects, policies, invariants, certificates, obstructions, quotients
10.2 The admissibility judgment as the atomic database record
10.3 Relational, graph, and vector layers
10.4 Retrieval of relevant policies and invariants
10.5 From formal corpus to machine-learnable admissibility

11. Training Mathematical AI on Certified Transformations

11.1 Training on X → Y is not enough
11.2 Training on X, Y, π, Iπ, C, and O
11.3 Contrastive examples: plausible but inadmissible transformations

11.4 Learning the weakest policy under which a claim becomes true
11.5 From theorem suggestion to proof-grounded admissibility assistant

12. Toward Future AI: From Mathematical Admissibility to General Trust

12.1 Mathematics as the clean first domain
12.2 Hallucination as policy failure
12.3 Abstraction as certified quotienting
12.4 Memory and reasoning as admissible continuity
12.5 The future AI slogan: intelligence is admissible transformation

13. References

Appendices

A. Minimal schema

B. Central machine-readable record

C. Minimal working prototype

D. First demonstration case

E. Public slogan

Art

1. The Problem: Mathematical AI Without Admissibility

1.1 Fluent theorem generation is not proof-grounded intelligence

Mathematical artificial intelligence is often judged by its ability to produce plausible definitions, conjectures, proofs, explanations, code, or formal scripts. This is understandable: fluency is visible, measurable, and immediately useful. A system that can state a theorem in polished language, suggest an argument, retrieve a relevant lemma, or write a Lean tactic appears to possess mathematical competence. Yet fluency is not the same as proof-grounded intelligence. A generated mathematical sentence may be grammatically perfect, stylistically convincing, and locally familiar while still failing to preserve the identity of the mathematical object under discussion. It may confuse equality with isomorphism, isomorphism with equivalence, equivalence with analogy, analogy with evidence, or evidence with proof. Such failures are not superficial errors of wording. They are failures of admissibility.

Mathematics is not merely a language game in which plausible continuations are rewarded. It is a discipline of permitted transformation. A proof is a chain of transformations licensed by definitions, axioms, inference rules, and previously established results. A simplification is allowed only if the simplified expression preserves the required value, structure, type, or proposition. A substitution is allowed only if the substituting object is identical, equivalent, isomorphic, definitionally equal, propositionally equal, or otherwise admissible in the context at hand. The meaning of “otherwise” is precisely the missing problem. The system must know which kind of sameness governs the transformation.

A mathematical AI that only generates plausible continuations is therefore operating below the level of mathematics. It may imitate mathematical discourse, but it does not yet control mathematical identity. In ordinary language, the difference between two near-synonymous expressions may be tolerated. In mathematics, small differences can be fatal. Replacing “for all” with “there exists,” a group isomorphism with a set bijection, a homotopy equivalence with a homeomorphism, or a natural transformation with a natural isomorphism can destroy the theorem. The issue is not that the generated sentence is ugly or awkward. The issue is that the transformation is not licensed.

This distinction is especially important for future mathematical AI because proof assistants, formal libraries, and neural theorem provers are converging. A model may retrieve formal facts, suggest tactics, generate proof sketches, and explain the result in natural language. But unless the system explicitly tracks admissibility, it can pass from one representation to another without knowing whether the passage preserves the relevant mathematical identity. The problem is not merely whether the proof checker accepts the final artifact. The deeper problem is whether the

system understands the policy under which each transformation is being made. Proof-grounded intelligence requires a control layer above fluency: the layer of admissible transformation.

1.2 Why plausibility is weaker than admissibility

Plausibility is a statistical, rhetorical, or heuristic property. A statement is plausible when it resembles true statements, fits a familiar pattern, satisfies local expectations, or appears compatible with surrounding context. Admissibility is different. A transformation is admissible when it is permitted under a specified policy of identity and supported by an appropriate certificate. The difference is decisive. Plausibility may guide search, but admissibility licenses substitution.

Consider two finite groups of the same cardinality. It is plausible to say they are “the same size.” Under a cardinality policy, that may be admissible. But it is not admissible to conclude that they are the same group. The group structure imposes stronger invariants: operation table, identity, inverses, element orders, subgroup lattice, commutativity, normal structure, representations, and other preserved features. A bijection of underlying sets is not a group isomorphism unless it preserves multiplication. Thus the same apparent comparison can be admissible under one policy and inadmissible under another.

This is the core reason plausibility is insufficient for mathematical AI. Neural systems are excellent at detecting patterns across examples. They can recognize that two statements are often associated, that one theorem resembles another, or that a proof strategy has worked nearby. Such recognition is valuable. It can propose candidates, retrieve relevant material, and guide exploration. But plausibility does not determine the policy of sameness. It does not by itself know whether the transformation preserves a required invariant. It does not establish a certificate. It does not reveal an obstruction. It does not distinguish a lawful quotient from a destructive collapse.

Mathematics depends on permitted loss as much as permitted preservation. Every abstraction forgets something. A quotient collapses distinctions. An isomorphism forgets presentation while preserving structure. A homotopy equivalence forgets pointwise rigidity while preserving deformable shape. A categorical equivalence forgets strict object-level equality while preserving recoverable categorical structure. A proof assistant may reduce expressions by definitional equality, rewrite by a theorem, transport along an equivalence, or simplify under a congruence. Each of these actions is plausible only because a deeper policy has already made it admissible.

A mathematical AI must therefore learn a hierarchy. At the lowest level, it may generate candidates. At the next level, it must identify the governing policy. Above that, it must identify the invariants required by the policy. Above that, it must construct or retrieve a certificate. Finally, it must either accept the transformation, reject it, mark it undecided, or suggest a

weaker policy under which the transformation becomes valid. This hierarchy converts plausibility into admissibility only when the missing proof of permission has been supplied.

The distinction also explains why near-misses are so dangerous. A generated statement can be close enough to sound correct while crossing an invisible mathematical boundary. “Equivalent categories are isomorphic” sounds close to a true idea but is generally false. “Bijective continuous maps are homeomorphisms” sounds plausible but fails without additional hypotheses. “Equal cardinality implies isomorphism” is false for most structured objects. These are not random mistakes. They are failures to respect the policy under which sameness is being asserted.

1.3 Hallucination as unauthorized mathematical substitution

In mathematical AI, hallucination should not be understood only as the production of nonexistent citations, false theorem names, or invalid proof steps. Those are symptoms. The deeper structure of hallucination is unauthorized substitution. The system replaces one object, claim, proof, theorem, condition, or equivalence with another without a valid admissibility certificate.

This definition is useful because it moves hallucination from psychology into algebra. The error is not merely that the model “made something up.” The error is that it performed a transformation $X \to Y$ while acting as if $X \equiv_\pi Y$, though no valid policy-indexed certificate C was available. In other words, the model silently asserted $\Gamma \vdash X \equiv_\pi Y : C$ without possessing C. Mathematical hallucination is certificate failure.

Examples are common. A model may replace a theorem with a nearby theorem whose hypotheses differ. It may cite compactness where completeness is required. It may assume commutativity in a noncommutative setting. It may treat an equivalence of categories as if it were equality of categories. It may transport a construction across an isomorphism but ignore the coherence data required for the transport. It may use a quotient while forgetting that the relation must be a congruence. It may summarize a proof while deleting the hypothesis on which the proof depends. In each case, the model has performed an unauthorized transformation of mathematical identity.

This view also explains why hallucination is not eliminated by retrieval alone. Retrieval can provide relevant material, but relevance is not admissibility. A retrieved theorem may be nearby but inapplicable. A lemma may resemble the needed lemma but require stronger assumptions. A proof may work in a commutative ring but fail in a semiring. A statement may hold for finite-dimensional vector spaces but not for infinite-dimensional modules. The admissibility question remains: under the current context Γ and policy π, does the retrieved object license the proposed substitution?

Nor is hallucination eliminated by formal proof checking alone, although proof checking is essential. A proof assistant can reject invalid formal scripts, but a mathematical AI also operates in the space before formalization: selecting definitions, choosing representations, proposing quotient structures, mapping informal claims into formal statements, retrieving lemmas, paraphrasing results, and explaining proofs. Hallucination can enter at any of these stages. A final checker can catch some failures, but an admissibility-aware system should prevent many failures upstream by tracking what kind of identity is being claimed.

The unauthorized-substitution view has another advantage: it treats hallucination and over-abstraction as the same family of errors. A hallucinated theorem invents a certificate. A bad abstraction erases a distinction without permission. A misleading analogy treats structural resemblance as if it were equivalence. A false proof sketch treats a plausible path as if it were a proof path. All are failures of policy-indexed identity. The remedy is not merely stricter output filtering. The remedy is a mathematical layer that asks, at every transformation: what is the policy, what invariant is preserved, and what certificate licenses the move?

1.4 The hidden identity problem inside AI reasoning

Every mathematical AI system contains an implicit theory of identity, even if it is never named. When a model retrieves a theorem, it decides that the retrieved theorem is relevantly the same kind of object as the current goal. When it rewrites an expression, it decides that the old and new expressions are interchangeable. When it summarizes a proof, it decides which details preserve the proof's identity and which can be omitted. When it translates informal mathematics into formal code, it decides how the informal statement is to be identified with a formal proposition. When it uses an embedding vector, it decides that semantic nearness is a meaningful proxy for mathematical relevance. These decisions are identity judgments.

The problem is that current systems often make these judgments implicitly. They do not expose the policy. They do not state whether sameness is literal equality, definitional equality, propositional equality, isomorphism, equivalence, homotopy equivalence, Morita equivalence, observational equivalence, syntactic similarity, semantic similarity, or mere analogy. They often collapse these modes into a single vague notion of "close enough." That is dangerous in mathematics, where the difference between modes of sameness is often the theorem itself.

Human mathematicians also use flexible identity, but they do so within disciplined traditions. We say two isomorphic groups are "the same" when the context concerns group-theoretic structure. We say two homeomorphic spaces are "the same" topological space when topology, not embedding, is at issue. We say equivalent categories are "the same" for many categorical purposes, while also knowing when strict equality matters. We work modulo definitional equality in type theory, modulo α-conversion in syntax, modulo gauge transformations in

physics, modulo homotopy in algebraic topology, and modulo equivalence relations in quotient constructions. Mathematical maturity consists partly in knowing which sameness relation is authorized.

Mathematical AI needs the same maturity, but it must be made explicit. The machine cannot merely inherit informal tolerance. It must represent the governing policy. It must know whether the current task permits replacement, transport, quotienting, normalization, or analogy. It must know whether the certificate is a proof term, an isomorphism, a homotopy, a universal property, a computation, a counterexample, a test, a typeclass instance, or a retrieved theorem. It must know when the certificate is missing.

The hidden identity problem is especially acute when AI moves between levels of representation. Informal mathematics is not identical to formal mathematics. A theorem statement in prose is not the same object as its Lean declaration. A diagram is not the same object as its categorical universal property. A proof sketch is not the same object as a proof term. A natural-language explanation is not the same object as the formal proof it describes. These objects may be connected by admissible transformations, but those transformations require policies. Without them, the system may confuse representation with object, notation with structure, analogy with equivalence, and explanation with proof.

Admissibility Algebra names this hidden layer. It says that every transformation made by mathematical AI should be understood as occurring under an explicit policy π. The system's primitive question should not be simply "what follows?" but "what may be replaced by what, under which policy, preserving which invariants, with which certificate?" This question exposes the identity structure already present inside mathematical reasoning.

1.5 From answer generation to certified transformation

The immediate goal is to shift mathematical AI from answer generation to certified transformation. An answer generator produces Y from X. A certified transformation system produces a judgment of the form $\Gamma \vdash X \equiv_\pi Y : C$, or else reports that the transformation is inadmissible or undecided. This changes the nature of the machine. It is no longer merely a device for producing mathematical-looking text. It becomes a system for proposing, checking, recording, and explaining permitted transformations.

The distinction can be expressed as a change in architecture. In a simple generative system, the pipeline is input → model → output. In an admissibility-aware system, the pipeline is context → policy selection → candidate transformation → invariant extraction → certificate search → verification → admissibility judgment → output. The output is no longer the primary artifact. The primary artifact is the certified judgment that licenses the output.

This matters because mathematics is cumulative. A false answer is bad, but a false transformation stored as usable knowledge is worse. If a system records an inadmissible equivalence, future reasoning can build on a corrupted identity. If a vector database retrieves analogous objects without labeling the policy under which they are comparable, downstream proof search may waste effort or produce invalid suggestions. If a theorem-proving assistant learns from successful-looking but uncertified transformations, it can internalize bad substitution habits. A mathematical AI corpus must therefore record not only objects and theorems, but admissibility judgments: what counted as the same, under which policy, with what certificate, and what failed.

Certified transformation also changes how failure is treated. In ordinary generation, failure often appears as the absence of a good output. In admissibility reasoning, failure can itself be informative. If X and Y are not the same under group isomorphism because an element-order invariant is violated, that obstruction is valuable knowledge. If two spaces are not homeomorphic but may be homotopy equivalent, the system has learned a policy boundary. If a statement fails under definitional equality but succeeds under propositional equality, the distinction guides formalization. If a quotient is invalid because the relation is not a congruence, the obstruction prevents destructive collapse. Mathematical AI should learn from certificates and obstructions alike.

The transition from generation to certification does not eliminate neural models. It repositions them. Neural systems remain useful for proposing transformations, retrieving candidates, guessing relevant policies, suggesting invariants, and searching proof space. But the admissibility layer governs whether those proposals may be accepted. The neural model becomes a source of candidates; the admissibility system becomes the authority over mathematical identity. This is not a rejection of machine learning. It is a discipline for machine learning in domains where identity is formal.

The first practical target is a mathematical admissibility assistant. Given two objects, claims, statements, or proof fragments, the assistant should identify the plausible policies under which they may be compared. It should say whether the objects are the same under literal equality, isomorphism, equivalence, homotopy equivalence, definitional equality, quotient equality, or another named policy. It should retrieve or construct the relevant certificate if the judgment is positive. It should identify the obstruction if the judgment is negative. It should distinguish inadmissibility from undecidability. It should suggest weaker or stronger policies when appropriate. Such a system would not need to solve all of mathematics to be valuable. It would already address one of the central weaknesses of current mathematical AI: uncontrolled sameness.

The long-term implication is broader. If mathematical AI can learn to certify transformations in the cleanest domain, then admissibility can later be exported to science, language, law, memory, and action. But mathematics should come first because its policies are already partially formalized. Equality, isomorphism, equivalence, quotienting, transport, rewriting, and proof checking provide a disciplined training ground. The first objective is therefore not artificial general intelligence, but a precise mathematical corpus in which identity is indexed by policy and certified by proof.

The problem of mathematical AI without admissibility can now be stated simply. A system that generates mathematical outputs without explicit policy-indexed identity does not yet know what it is allowed to treat as the same. It may be useful, but it is not secure. It may be fluent, but it is not proof-grounded. The required next step is an algebra of admissible transformation: a mathematics of what may count as the same, under which policy, with which certificate.

2. Algebra, Abstract Algebra, and the Question of Sameness

2.1 Algebra as lawful operation

Algebra begins with lawful operation. Its first concern is not identity in the modern structural sense, but manipulation under rules. Quantities are added, multiplied, factored, inverted, substituted, eliminated, solved, and transformed. Equations are rearranged without changing their solution sets. Expressions are simplified without changing their values. Unknowns are isolated. Polynomials are decomposed. Systems are reduced. Algebra, in this initial sense, is the discipline of permitted calculation.

This already contains the seed of admissibility. When a student solves an equation, every step is a claim that one expression may be replaced by another without changing the relevant mathematical object. Adding the same quantity to both sides, multiplying by a nonzero scalar, expanding a product, factoring a polynomial, or substituting an equivalent expression are not arbitrary operations. They are licensed moves. The identity preserved may be a number, a solution set, a function, an equation, a relation, or an expression under interpretation. Even elementary algebra is therefore not mere symbol pushing. It is symbol transformation under permission.

The permission is often hidden because the rules are familiar. We say that $x + 3 = 7$ may be transformed into $x = 4$. But this transformation is not a metaphysical equality between inscriptions. It is an admissible passage between equations preserving a solution. We say that $x^2 - 1$ may be replaced by $(x - 1)(x + 1)$. But that replacement is governed by a polynomial identity. We say that a rational expression may be simplified by canceling a common factor, but only

where the canceled factor is nonzero. The operation appears mechanical only after its admissibility conditions have been absorbed.

Algebra therefore trains mathematical thought in the habit of lawful transformation. The central practical question is: what may I do to this expression while preserving what matters? In elementary contexts, what matters may be numerical value, equation solution, or symbolic identity. In later contexts, what matters becomes structure, morphism, invariant, equivalence class, or proof object. The continuity is important. Admissibility Algebra does not reject algebra. It makes explicit the permission structure already implicit in algebraic practice.

The limitation of elementary algebra is that its transformations are usually local, syntactic, and computational. They operate inside a fixed background where the objects and rules are presumed. The student need not ask what kind of sameness is being used; the textbook has already selected the policy. But as mathematics advances, the background becomes plural. The same object can be treated syntactically, semantically, algebraically, geometrically, categorically, homotopically, or computationally. Each treatment licenses different transformations. Algebra teaches lawful operation; Admissibility Algebra asks which law is governing the operation.

This matters for mathematical AI because current systems often imitate the surface of algebraic transformation without representing the policy that licenses it. They may simplify, rewrite, substitute, or analogize, but they do not always know which invariant is supposed to survive. Algebra's original lesson is that transformations are never merely movements of symbols. They are lawful movements under a preservation requirement. Future mathematical AI must recover that lesson at scale.

2.2 Abstract algebra as lawful structure

Abstract algebra generalizes algebra by shifting attention from particular calculations to the structures that make calculations lawful. Instead of asking only how numbers combine, it asks what kind of system permits a given pattern of combination. Groups, rings, fields, modules, vector spaces, lattices, algebras, and related structures are not merely collections of elements. They are domains of operation constrained by axioms. The operations are meaningful because they preserve and express structure.

This is a decisive abstraction. In elementary algebra, the rules may appear attached to familiar objects: integers, real numbers, polynomials, matrices. In abstract algebra, the rules become portable. A group may consist of symmetries, permutations, matrices, residues, paths, transformations, or automorphisms. What matters is not the material presentation of the elements but the lawful pattern of composition, identity, and inverse. A ring may be numbers, functions, polynomials, operators, or cohomology classes. A module may generalize vector-space reasoning beyond fields. Structure becomes the real object of study.

With this shift, sameness also changes. Two groups may have different elements, names, encodings, or origins, yet be the same as groups if there is an isomorphism between them. The isomorphism certifies that multiplication, identity, and inverses are preserved. Two vector spaces over the same field may be identified up to dimension. Two finite cyclic groups of the same order may be treated as structurally the same even if their elements are presented differently. Abstract algebra thus replaces surface identity with structural identity.

This is the birthplace of modern admissibility thinking. An isomorphism is not merely a convenient comparison. It is a certificate of permitted replacement under a structural policy. If G and H are isomorphic as groups, then group-theoretic statements true of G may be transported to H. But the transport is scoped. The isomorphism licenses replacement for group-theoretic purposes, not for every possible purpose. If G is realized as rotations of a square and H as residue classes modulo 4, the group structure may be the same while geometric or arithmetic meanings differ. The policy matters.

Abstract algebra therefore reveals that sameness is indexed by structure. The same underlying set may carry different group structures. The same group may be represented by different presentations. The same ring may be viewed as an additive group, a multiplicative monoid with zero, a module over itself, a category with one object, or an algebra over a base ring. Each view selects different admissible transformations. A bijection of sets may be insufficient for group identity. A group isomorphism may be insufficient for ring identity. A ring isomorphism may be insufficient for ordered-ring identity. Sameness strengthens or weakens according to the policy of structure being preserved.

This is essential for mathematical AI. A model cannot safely infer sameness from superficial resemblance. It must know which structure is being preserved. If the task concerns cardinality, a bijection may suffice. If it concerns group theory, the operation must be preserved. If it concerns topology, open sets or continuous structure must be preserved. If it concerns category theory, functorial and natural data may be required. Abstract algebra teaches that identity is not absolute presentation. It is structure-preserving admissibility.

Yet abstract algebra still does not fully answer who selects the structure, how policies compare, when weaker identity is acceptable, when quotienting destroys information, or how AI should record and learn from admissible and inadmissible substitutions. It gives us many policies of sameness, especially isomorphism, but it does not make policy itself the central object. Admissibility Algebra begins precisely there.

2.3 Morphisms as structure-preserving transformations

The central invention of modern structural mathematics is the morphism. A morphism is not merely a function between sets. It is a transformation that respects the structure of its domain

and codomain. Group homomorphisms preserve multiplication. Ring homomorphisms preserve addition and multiplication, subject to convention. Linear maps preserve vector addition and scalar multiplication. Continuous maps preserve topological structure. Smooth maps preserve differentiable structure. Functors preserve categorical composition and identities. Natural transformations preserve functorial structure across a whole category of objects.

Morphisms make structure mobile. They allow one object to be related to another without requiring literal sameness. A homomorphism may forget information, embed information, project information, quotient information, or preserve enough information to form an isomorphism. A morphism is therefore a controlled passage between objects. It tells us not only that two things are related, but what kind of structure survives the relation.

This is already close to the admissibility viewpoint. A morphism can be interpreted as a candidate transformation. Whether it licenses identity depends on the policy. A general group homomorphism does not make two groups the same. A monomorphism may identify one group as a substructure of another. An epimorphism may express a quotient or projection. An isomorphism permits structural replacement. Thus not every morphism is an admissibility certificate for sameness. Morphisms exist in many strengths, and identity requires the right kind of morphism under the right policy.

For example, a map $f : G \to H$ that preserves group multiplication provides information about the relationship between G and H. If f is injective, G is represented faithfully inside H. If f is surjective, H is a quotient image of G. If f is bijective with structure-preserving inverse, then G and H are isomorphic. These distinctions are not decorative. They determine what can be substituted for what. A quotient map may allow passage from detailed structure to coarse structure, but it does not allow reconstruction unless extra data exist. An embedding may allow local identification with a subobject, but not global identity with the codomain. An isomorphism provides the certificate needed for structural sameness.

Category theory radicalizes this insight by making morphisms primary. Objects are understood through their relations, maps, universal properties, and functorial behavior. This does not eliminate identity; it relocates identity into patterns of admissible transformation. The Yoneda perspective, for example, says that an object is determined by how other objects map into it. Universal properties define objects up to unique isomorphism. Adjunctions, equivalences, limits, colimits, and representability all express identity through structured relations rather than isolated substance.

For mathematical AI, morphisms provide a natural grammar of transformation. A system trying to reason mathematically should not merely store terms and theorem statements. It should store typed transformations: homomorphisms, embeddings, quotient maps, equivalences,

functors, natural transformations, rewrites, reductions, transports, and proof steps. Each transformation should be indexed by the structure it preserves and the policy under which it may license substitution.

This is where many AI errors can be reclassified. The model has often found a morphism-like relation but mistaken its strength. It treats analogy as homomorphism, homomorphism as isomorphism, equivalence as equality, or quotient as reversible compression. The remedy is not simply more examples. The remedy is an explicit admissibility hierarchy: which morphisms exist, which invariants they preserve, which certificates they provide, and which substitutions they license.

2.4 Equivalence beyond literal equality

Mathematics repeatedly discovers that literal equality is too rigid. Two expressions may differ syntactically while denoting the same value. Two groups may have different elements while sharing the same group structure. Two topological spaces may be different as sets but homeomorphic. Two spaces may fail to be homeomorphic yet share the same homotopy type. Two categories may fail to be isomorphic yet be equivalent. Two type-theoretic structures may be equivalent in a way that, under univalence, becomes a legitimate form of identity. Modern mathematics advances by refining the meaning of "same."

Literal equality remains indispensable. It provides the strongest and most rigid identity relation. But mathematical practice constantly introduces weaker, more flexible, and more context-sensitive forms of sameness. Equality of elements, equality of sets, equality of functions, equality of structures, isomorphism, equivalence, homotopy equivalence, weak equivalence, Morita equivalence, gauge equivalence, observational equivalence, and definitional equality all serve different purposes. Each permits some transformations and forbids others.

The crucial point is that these modes are not interchangeable. Treating isomorphism as equality may be harmless in one context and disastrous in another. Treating equivalent categories as identical may be productive for categorical reasoning but invalid if strict equality of objects or chosen representatives matters. Treating homotopy equivalent spaces as the same may preserve homotopy invariants but destroy point-set topological information. Treating gauge-equivalent fields as physically identical may preserve observables while ignoring representational degrees of freedom. Every equivalence forgets something. The question is whether the forgotten distinction is inadmissible noise or essential structure.

Admissibility Algebra turns this observation into a formal principle. An equivalence relation, by itself, does not automatically authorize every substitution. It must be paired with a policy specifying the context of use, preserved invariants, permitted quotienting, certificate type, and substitution scope. Two objects may be equivalent under π_1 but not under π_2. A transformation

may be admissible for computation but not for explanation, admissible for classification but not for reconstruction, admissible for homotopy theory but not for geometry, admissible for theorem search but not for proof export.

This matters especially in mathematical AI because models often blur equivalence levels. A system trained on broad mathematical text may learn that mathematicians say "the same" in many contexts. But human mathematicians rely on tacit discipline to know which sameness is intended. AI systems must make this discipline explicit. They must record whether a claim is being made up to equality, up to isomorphism, up to equivalence, up to homotopy, up to definitional reduction, or up to a named quotient. Without the subscript π, the symbol of sameness is underdetermined.

Equivalence beyond literal equality is not a weakness of mathematics. It is one of its great strengths. It allows mathematics to ignore irrelevant presentation and preserve essential structure. It enables classification, abstraction, transport, and generality. But the power of equivalence depends on knowing its jurisdiction. An equivalence that is too weak may erase the theorem. An equivalence that is too strong may block useful abstraction. A policy-indexed system can navigate this by asking: what kind of sameness is required here?

Mathematical AI must therefore learn equivalence as a family of governed permissions, not as a vague semantic similarity. The appropriate output is not simply "X and Y are equivalent," but "X and Y are equivalent under policy π, preserving invariants Iπ, with certificate C, and substitution is licensed only in contexts compatible with π." This is the bridge from abstract mathematics to admissibility-aware computation.

2.5 Why mathematical AI needs the next layer

Algebra teaches lawful operation. Abstract algebra teaches lawful structure. Category theory teaches lawful relation. Homotopy type theory teaches that equivalence can have identity-like force. But mathematical AI requires one more explicit layer: lawful admission. The system must know not only what transformations exist and what structures they preserve, but whether a given transformation is authorized to count as sameness in the current context.

This is the next layer because AI systems operate across boundaries. They move between informal and formal language, between theorem statements and proof scripts, between semantic retrieval and logical inference, between examples and conjectures, between diagrams and definitions, between human explanations and machine-checkable artifacts. Each boundary crossing is a transformation. Each transformation requires a policy. Without an admissibility layer, the system may cross boundaries by resemblance, proximity, or convenience rather than permission.

Mathematical AI also needs the next layer because it must manage multiple strengths of identity simultaneously. In a single task, a system may need literal equality for computation, isomorphism for classification, equivalence for categorical transport, definitional equality for proof checking, and semantic similarity for retrieval. These are not conflicting if they are indexed and scoped. They become dangerous only when collapsed into one undifferentiated notion of sameness. Admissibility Algebra supplies the indexing.

The next layer is also necessary for learning. A model trained only on correct outputs learns correlations between prompts and answers. A model trained on proof scripts learns patterns of formal derivation. But a model trained on admissibility judgments learns something deeper: which transformations are licensed under which policies and why. It learns that the same pair of objects can have different sameness statuses depending on the policy. It learns to distinguish certificate from plausibility and obstruction from failure to search. It learns that a quotient is not merely compression but controlled collapse.

The database required for such a system is therefore not just a theorem database. It is an admissibility database. Its atomic record is not merely "theorem T is true," but "in context Γ, object X counts as the same as object Y under policy π with certificate C," or else "the proposed identification fails because obstruction O violates invariant I." Such a database would allow mathematical AI to retrieve not only facts, but permissions. It would allow the system to ask which substitution is licensed before using it.

This next layer does not replace existing formal systems. It organizes them. Proof assistants already enforce many forms of admissibility locally through type checking, definitional equality, rewrite rules, tactic validation, and proof-term construction. Category theory and HoTT already provide deep theories of equivalence and transport. Abstract algebra already supplies a rich taxonomy of structure-preserving maps. Admissibility Algebra adds a unifying index: the policy π under which a transformation is permitted to count as identity for a specified purpose.

The need can be summarized directly. Mathematical AI will fail whenever it treats mathematical resemblance as mathematical identity. It will become more reliable when it learns that identity is policy-indexed, certificate-bearing, invariant-preserving, and substitution-scoped. This is the reason Admissibility Algebra is not merely a philosophical gloss on existing mathematics. It is the missing control layer for mathematical AI: the layer that tells the machine what may count as the same.

3. The Primitive Judgment of Admissibility Algebra

3.1 The judgment Γ ⊢ X ≡π Y : C

The primitive judgment of Admissibility Algebra is Γ ⊢ X ≡π Y : C. It asserts that, in context Γ, objects X and Y count as the same under admissibility policy π with certificate C. This is not merely notation. It is the central shift from unindexed sameness to governed sameness. The judgment does not say simply that X equals Y, nor merely that X resembles Y, nor even that X is equivalent to Y in some unspecified way. It says that a specified policy π authorizes the identification, and that the authorization is witnessed by a certificate C.

Each part of the judgment is essential. Γ supplies the mathematical environment: definitions, assumptions, universe levels, ambient category, type theory, algebraic structure, theorem dependencies, allowed rewrite rules, and intended domain. X and Y are the objects, terms, propositions, structures, diagrams, proofs, models, or representations being compared. π is the policy of sameness: equality, isomorphism, homotopy equivalence, categorical equivalence, definitional equality, quotient equivalence, observational equivalence, or another admissibility rule. C is the evidence that the comparison is valid under that policy.

The subscript π is the key innovation. Ordinary mathematical notation often suppresses policy because humans infer it from context. We say that two finite-dimensional vector spaces over a field are “the same” when they have the same dimension. We say that two groups are “the same” when they are isomorphic. We say that two categories are “the same” when they are equivalent. We say that two gauge potentials represent the same physical state when they differ by a gauge transformation. In each case, sameness has a hidden subscript. Admissibility Algebra brings that subscript into the judgment.

The certificate C prevents the judgment from becoming a declaration by preference. The claim X ≡π Y is not accepted because it sounds reasonable. It is accepted because a valid certificate exists. For equality, C may be a proof of equality. For group isomorphism, C may be a bijective homomorphism with inverse or proof of bijectivity. For categorical equivalence, C may be functors with natural isomorphisms satisfying coherence. For homotopy equivalence, C may be maps and homotopies witnessing deformation equivalence. For a quotient relation, C may be membership in the same equivalence class. For a proof assistant, C may be a formal proof object accepted by the kernel.

The primitive judgment can also fail. In that case, the system may record Γ ⊬ X ≡π Y, preferably with an obstruction Oπ(X,Y). The obstruction is not merely an absence of proof. It may be a preserved invariant that differs between X and Y, a failed hypothesis, an invalid relation, a missing coherence condition, a counterexample, a type mismatch, or an impossible transport. Positive certificates and negative obstructions together define the admissibility boundary.

This makes the judgment directly useful for mathematical AI. A generative model proposes Y from X. An admissibility system asks whether $\Gamma \vdash X \equiv_\pi Y : C$ can be established. If yes, the transformation is licensed. If no, it must be rejected, qualified, weakened, or sent to another policy. The output of the AI is therefore not merely Y. The correct output is the certified admissibility judgment that explains whether Y may stand in for X, and under what conditions.

3.2 Context Γ as mathematical jurisdiction

The context Γ is the jurisdiction within which admissibility is judged. It contains the assumptions, definitions, structures, ambient theories, declared variables, type constraints, universe levels, hypotheses, available theorems, and intended interpretation that determine what transformations are lawful. Without Γ, no admissibility judgment is fully meaningful. The same pair X and Y may count as the same in one context and not in another.

In elementary algebra, Γ may contain a field assumption, a nonzero condition, or a domain restriction. The expression $(x^2 - 1)/(x - 1)$ may be simplified to $x + 1$ only under a context excluding $x = 1$. Without that exclusion, the transformation changes the domain of definition. The simplification is plausible but not globally admissible. The context is not an afterthought; it is the jurisdiction determining whether cancellation is lawful.

In abstract algebra, Γ may specify whether objects are sets, groups, rings, ordered rings, topological groups, Lie groups, modules, or categories. A bijection may be admissible under the context of finite sets but insufficient under the context of groups. A group isomorphism may be insufficient under the context of topological groups unless continuity and continuous inverse are included. A ring isomorphism may fail to preserve order if the context is ordered algebra. Thus Γ determines which structures are visible and which invariants must be preserved.

In type theory and proof assistants, Γ has an even more literal status. It is the typing context: variables, hypotheses, definitions, instances, universe parameters, and local assumptions. A term may type-check under one context and fail under another. A rewrite may be valid only after an instance is available. A theorem may apply only when typeclass conditions have been resolved. The proof kernel enforces admissibility relative to Γ, although it may not name the broader policy explicitly.

The word jurisdiction is deliberate. A context authorizes some moves and forbids others. It determines which court of mathematical law is hearing the case. The same inscription may be interpreted differently in set theory, category theory, homotopy type theory, constructive mathematics, classical mathematics, algebraic geometry, differential geometry, or formalized Lean code. The context selects the background logic and the admissibility standards.

For mathematical AI, failure to track Γ is a primary source of error. A model may retrieve a theorem whose statement resembles the goal but belongs to a different context. It may apply a

result from finite-dimensional linear algebra to an infinite-dimensional setting. It may use classical reasoning in a constructive context. It may invoke commutativity where only associativity is given. It may treat a category as small when size matters. These are not merely missing details. They are jurisdictional failures.

An admissibility database must therefore store Γ as part of the atomic record. It is not enough to store that X and Y are equivalent. One must store where, under what assumptions, in which formal system, relative to which definitions, and under which policy. The same theorem name, informal phrase, or mathematical object may have different admissibility behavior in different contexts. Γ is what prevents the system from treating mathematics as context-free symbol association.

3.3 Objects X and Y as presentations, structures, or claims

The terms X and Y in $\Gamma \vdash X \equiv_\pi Y : C$ should be interpreted broadly. They may be mathematical objects, but they may also be presentations, structures, statements, proof fragments, diagrams, algorithms, models, definitions, normal forms, or formal encodings. Admissibility Algebra is not limited to comparing two already well-defined abstract objects. It governs any proposed transformation in mathematical reasoning.

In one case, X and Y may be two presentations of a group. One presentation may use generators and relations; another may use permutations; another may use matrices; another may use residue classes. The question is not whether the presentations are literally identical. The question is whether they present the same group under the selected policy, and whether there is a certificate, such as an explicit isomorphism or proof of a universal presentation property.

In another case, X and Y may be two theorem statements. An informal theorem in prose may be transformed into a formal Lean proposition. This transformation is not equality in the ordinary sense. It is an admissibility claim: the formal proposition preserves the mathematical content of the informal statement under a formalization policy. The certificate may involve definitions, examples, expert annotation, proof alignment, and eventual verification. If a hypothesis is omitted, generalized, narrowed, or mistranslated, the transformation may fail.

X and Y may also be proof fragments. A human proof sketch may be transformed into a formal proof script. A tactic script may be transformed into a proof term. A proof term may be transformed into a natural-language explanation. Each transformation changes representation. The admissibility policy determines what must be preserved: theorem conclusion, hypotheses, dependencies, constructive content, computational behavior, proof irrelevance, readability, or kernel-checkable validity.

In computational mathematics, X and Y may be algorithms. A slow algorithm may be replaced by a faster one under a behavior-preservation policy if they compute the same function on the

relevant domain. A symbolic expression may be replaced by a numerical approximation under an approximation policy if error bounds are certified. A matrix representation may be replaced by a sparse representation if linear action is preserved. These cases show that admissibility is not only about abstract equality; it is about permitted transformation under intended use.

In mathematical AI, X is often the user's intended object and Y is the system's generated substitute. If the AI summarizes a theorem, Y is a compressed representation of X. If it retrieves a lemma, Y is a candidate support for X. If it suggests a proof, Y is a proposed derivation of X. If it translates mathematics into formal code, Y is a formal representation of X. If it performs analogy, Y may be an object from another domain. In all cases, the system should ask whether Y is admissible as a replacement, explanation, support, or transformation of X under policy π.

This broad interpretation makes the primitive judgment useful across the entire mathematical AI pipeline. It does not merely compare final objects. It governs every intermediate passage. A model output is not accepted because it is fluent. A retrieved theorem is not accepted because it is nearby. A formalization is not accepted because it compiles. A summary is not accepted because it is shorter. Each must be judged as a transformation between X and Y under a policy with a certificate.

3.4 Policy π as the rule of permitted sameness

The admissibility policy π is the rule that determines what kind of sameness is being asserted. It is the answer to the question: same for what purpose, under what invariants, with what allowed transformations, and by what kind of certificate? Without π, the assertion X ≡ Y is incomplete. It may mean literal equality, definitional equality, propositional equality, isomorphism, equivalence, weak equivalence, homotopy equivalence, quotient equivalence, observational equivalence, semantic equivalence, or something weaker such as analogy. These are different judgments.

A policy is not merely a label. It includes a domain of application, a relation or class of transformations, required invariants, admissible certificates, coherence conditions, and substitution scope. For group isomorphism, the policy requires preservation of group multiplication, identity, and inverses by a bijective structure-preserving map. For ring isomorphism, it requires preservation of ring operations and units according to convention. For homeomorphism, it requires continuous maps with continuous inverse. For homotopy equivalence, it requires maps whose composites are homotopic to identities. For categorical equivalence, it requires functorial data and natural isomorphism or equivalent formulations.

Policies may be ordered by strength. Literal equality is usually stronger than isomorphism. Isomorphism is usually stronger than equivalence of categories. Homeomorphism is stronger than homotopy equivalence. Definitional equality is stronger than propositional equality in

many type-theoretic settings. A stronger policy preserves more information and permits fewer identifications. A weaker policy collapses more distinctions and enables broader abstraction. Neither strength nor weakness is intrinsically superior. The correct policy depends on the context and purpose.

Policies may also refine one another. A topological group isomorphism refines group isomorphism by adding topological preservation. An ordered ring isomorphism refines ring isomorphism by preserving order. A unitary equivalence refines Hilbert-space isomorphism by preserving inner-product structure. A formalization policy may refine semantic preservation by requiring type-correct, proof-checkable representation. Admissibility Algebra should record these refinements explicitly, because they determine whether a certificate under one policy can be reused under another.

The policy π also governs substitution. If $X \equiv_\pi Y$, then replacement is allowed only in contexts compatible with π. If two groups are isomorphic, group-theoretic properties may transport, but properties depending on a chosen presentation may not. If two spaces are homotopy equivalent, homotopy invariants transport, but point-set topological properties need not. If two categories are equivalent, many categorical properties transport, but strict object equality does not. Thus π determines not only whether X and Y count as the same, but where that sameness may be used.

For mathematical AI, policy selection is a core cognitive task. The system must infer whether the user or proof state requires equality, isomorphism, equivalence, approximation, quotienting, or analogy. It must not silently substitute a weaker policy for a stronger one. It must not reject a valid abstraction by demanding unnecessary literal equality. It must not treat semantic similarity as mathematical equivalence. It must expose the policy: “under cardinality these objects agree; under group isomorphism they do not.” This exposure is the beginning of trustworthy mathematical reasoning.

3.5 Certificate C as the witness of admissible identity

The certificate C is the witness that the policy-indexed identity has been established. Without C, the judgment $\Gamma \vdash X \equiv_\pi Y$ remains an assertion. With C, it becomes a certified transformation. The nature of C depends on π. There is no universal certificate type. Admissibility Algebra therefore treats certificates as policy-indexed objects: $C_\pi(X,Y)$ is the type or class of valid witnesses that X and Y count as the same under π.

For literal equality, C may be a proof that $X = Y$. For definitional equality, C may be normalization or kernel reduction. For propositional equality, C may be a proof term. For group isomorphism, C may be an explicit isomorphism $\phi : G \to H$ with proof that ϕ preserves multiplication and has inverse. For categorical equivalence, C may be functors $F : C \to D$ and $G : D \to C$ with natural

isomorphisms IdC ≅ G∘F and F∘G ≅ IdD, plus coherence as required. For homotopy equivalence, C may be maps f and g with homotopies g∘f ~ id and f∘g ~ id. For quotient equality, C may be a proof that two representatives are related by the quotient relation.

The certificate is not the same as a verbal explanation. An explanation may describe why a judgment is true, but a certificate is the mathematical witness that makes it true under the policy. In a proof assistant, the distinction is sharp: the proof object or kernel-accepted term is the certificate; prose around it is commentary. In informal mathematics, the certificate may be less syntactically rigid but still has mathematical form: a constructed map, a diagram chase, a universal property, a counterexample, a computation, or an invariant comparison.

Certificates may be constructive or nonconstructive, explicit or implicit, canonical or noncanonical. These distinctions matter. An existence proof of an isomorphism may certify structural sameness, but an explicit isomorphism may be needed for computation or transport. A noncanonical equivalence may license classification but not a preferred representation. A proof of mere existence may be insufficient for an algorithm requiring a witness. Thus certificate strength is part of admissibility.

Certificates must also compose. If $\Gamma \vdash X \equiv_\pi Y : C_1$ and $\Gamma \vdash Y \equiv_\pi Z : C_2$, then under many policies there should be a composed certificate witnessing $\Gamma \vdash X \equiv_\pi Z$. But composition may require coherence. In category theory and homotopy type theory, coherence cannot be ignored. A future admissibility system must therefore record not only isolated certificates but their composition laws, identity certificates, inverses when present, and compatibility with substitution.

For mathematical AI, certificate retrieval and construction are the heart of trust. A model may propose that two objects are equivalent. The admissibility system must ask: where is the certificate? Can it be constructed, retrieved, checked, or reduced to known certificates? Is it formal or informal? Does it satisfy the policy? Does it preserve the declared invariants? If not, the model should not present the transformation as established.

The certificate also makes learning possible. A training corpus containing only true statements teaches association. A corpus containing certificates teaches permission. The model learns not just that X and Y are related, but how they are related and what kind of evidence licenses their identification. This is why Admissibility Algebra is naturally suited to proof-grounded mathematical AI. It converts mathematical sameness into machine-learnable certified records.

The primitive judgment can now be read in full. Γ names the jurisdiction. X and Y name the objects or representations being compared. π names the rule of permitted sameness. C names the witness. Together they form the atomic unit of admissible mathematical transformation. A mathematical AI built around this judgment does not merely generate conclusions. It records,

verifies, and learns the conditions under which one mathematical form may legitimately stand in for another.

4. Policies as First-Class Mathematical Objects

4.1 Why equivalence is not self-authorizing

Equivalence is one of the great liberating ideas of modern mathematics, but an equivalence does not authorize itself. To say that two objects are equivalent is still incomplete unless one specifies the domain, purpose, invariants, certificate type, and substitution scope of the equivalence. Equivalence is not a magic word that collapses all distinctions. It is a disciplined permission to ignore some differences while preserving others. The discipline lies in the policy.

A set bijection does not authorize group-theoretic replacement. A group isomorphism does not authorize replacement as ordered groups unless order is preserved. A ring isomorphism does not authorize replacement as topological rings unless the topology is respected. A homotopy equivalence does not authorize replacement as homeomorphic spaces. An equivalence of categories does not authorize strict equality of objects. Each equivalence has jurisdiction, and outside that jurisdiction it may fail to preserve what matters.

This is why Admissibility Algebra treats equivalence as policy-indexed. The claim $X \equiv Y$ is underspecified. The correct claim is $X \equiv_\pi Y$, where π states what sort of identity is being asserted. Without π, the system does not know whether it is allowed to rewrite, transport, quotient, classify, compute, explain, or merely compare. A mathematical AI that sees "equivalent" without policy may behave as if all equivalences license all replacements. That is precisely the error Admissibility Algebra is designed to prevent.

Equivalence also requires authority. The authority may be an axiom, definition, theorem, construction, proof object, universal property, formal kernel, or established mathematical convention. An equivalence relation on a set authorizes quotienting only if it is the relevant relation for the intended construction. A congruence on an algebraic structure authorizes quotient algebra formation because it respects the operations. A mere relation may partition a set but fail to preserve the structure required for an algebraic quotient. The difference is not cosmetic. It is the difference between lawful collapse and destructive collapse.

Equivalence is therefore not self-authorizing because it always answers a prior question: equivalent with respect to what? The mathematical object itself does not decide this alone. The ambient context Γ and policy π determine which features count as invariant. Two objects may be equivalent under one policy, inequivalent under another, and incomparable under a third. The existence of one equivalence does not imply universal substitutability.

This point is essential for mathematical AI. A neural model may find a nearby theorem, a similar proof pattern, or an analogy between structures. But similarity is not equivalence, and equivalence is not admissibility unless the relevant policy is supplied. The AI must not treat a recognized pattern as permission. It must ask whether the pattern corresponds to a certified relation under the governing policy. This is the difference between associative intelligence and proof-grounded mathematical intelligence.

In human mathematical practice, policy is often tacit because experts know the domain. When a category theorist says two categories are “the same,” the intended policy may be equivalence of categories, not isomorphism. When an algebraic topologist says two spaces are “the same,” the intended policy may be weak homotopy equivalence, homotopy equivalence, homeomorphism, or diffeomorphism depending on context. When a proof assistant accepts a rewrite, the intended policy may be definitional equality or a declared theorem. Humans infer these policies from training. Machines must represent them explicitly.

Thus the first principle of policy algebra is: no equivalence licenses substitution until its admissibility policy has been declared or inferred and certified. Equivalence by itself describes a relation. Policy determines what that relation permits.

4.2 Equality, isomorphism, equivalence, and homotopy equivalence as policies

The familiar hierarchy of mathematical sameness can be reinterpreted as a hierarchy of admissibility policies. Literal equality, isomorphism, categorical equivalence, and homotopy equivalence are not merely different words for “same.” They are different policies for deciding when replacement is permitted.

Literal equality is the strongest and most rigid policy. It identifies objects without changing presentation, except insofar as the underlying formal system already treats expressions as equal. In proof assistants, literal equality may appear as propositional equality or definitional equality depending on the system and context. Definitional equality is computationally recognized by reduction. Propositional equality requires a proof. Both are strict compared with most structural policies. Their advantage is maximal substitutability. Their disadvantage is inflexibility.

Isomorphism weakens literal equality by permitting different presentations of the same structure. Two groups may be isomorphic though their elements are named differently. Two vector spaces may be isomorphic though their vectors are represented differently. Two graphs, rings, modules, or topological spaces may be isomorphic in the relevant structural category. Isomorphism preserves structure by a reversible map. It permits replacement for properties invariant under that structure. As a policy, isomorphism says: presentation may change, but recoverable structure must not.

Categorical equivalence weakens strict category isomorphism. Two categories may fail to be isomorphic as categories yet be equivalent in the sense that they have the same categorical content up to functors and natural isomorphisms. This policy is powerful because categories often contain redundant choices of representatives. Equivalence permits one to ignore inessential strictness while preserving the behavior of objects and morphisms up to coherent comparison. But it does not license every strict claim. Object equality, chosen representatives, and definitional features may not be preserved.

Homotopy equivalence weakens homeomorphism by permitting continuous deformation. Two spaces may not be the same as point-set topological spaces, yet may have the same homotopy type. This policy preserves homotopy-invariant features while discarding finer topological data. It is therefore appropriate in algebraic topology when the intended invariants are fundamental groups, homology, cohomology, homotopy groups, and related deformational structures. It is not appropriate when exact embeddings, local topology, smooth structure, or metric properties matter.

Each of these policies has a characteristic certificate. Equality requires a proof of equality or reduction. Isomorphism requires invertible structure-preserving maps. Categorical equivalence requires functors and natural isomorphism data or equivalent full-faithful-essentially-surjective data. Homotopy equivalence requires maps and homotopies connecting composites to identities. These certificates are not interchangeable. A homotopy equivalence is not a homeomorphism certificate. A bijection is not a group isomorphism certificate. A semantic resemblance is not an equality proof.

These policies also determine different quotient worlds. Under equality, distinctions are maximally preserved. Under isomorphism, presentations collapse into structural types. Under categorical equivalence, strict categorical presentations collapse into equivalent categorical content. Under homotopy equivalence, spaces collapse into homotopy types. Each quotient is useful because it forgets the right distinctions for a chosen purpose. Each is dangerous if used outside that purpose.

A mathematical AI must therefore treat these modes as named policies rather than fuzzy synonyms. A statement such as “X and Y are the same” should trigger policy resolution. Are they literally equal? Isomorphic? Equivalent as categories? Homotopy equivalent? Equal in a quotient? Definitionally reducible? Propositionally equal? Merely analogous? The system should not answer with a single undifferentiated yes or no. It should identify the policy under which yes or no is meaningful.

This provides a practical training task. Given two objects, ask the AI to evaluate them under multiple policies. For example, two finite groups may be the same under cardinality but not

under group isomorphism. Two categories may be equivalent but not isomorphic. Two spaces may be homotopy equivalent but not homeomorphic. Two formal expressions may be propositionally equal but not definitionally equal. Such examples train the model to respect the policy hierarchy.

The result is a controlled pluralism of sameness. Mathematics does not have one identity relation in practice. It has many disciplined identity policies. Admissibility Algebra does not abolish these; it indexes them, compares them, certifies them, and makes them machine-readable.

4.3 Policy strength, weakness, refinement, and failure

Policies can be compared by strength. A stronger policy preserves more structure and permits fewer identifications. A weaker policy preserves less structure and permits more identifications. Literal equality is stronger than isomorphism. Isomorphism is often stronger than categorical equivalence. Homeomorphism is stronger than homotopy equivalence. Ordered-ring isomorphism is stronger than ring isomorphism. Topological-group isomorphism is stronger than abstract group isomorphism. Definitional equality is stronger than propositional equality in systems where definitional equality is computationally stricter.

Strength is not superiority. A strong policy may be too rigid for classification. A weak policy may be too coarse for computation. The correct policy is the one that preserves the invariants relevant to the task. If the goal is to classify finite-dimensional vector spaces over a fixed field, dimension may be sufficient. If the goal is to compute with a chosen basis, mere isomorphism may erase useful data. If the goal is homotopy classification, homeomorphism may preserve too much. If the goal is geometric modeling, homotopy equivalence may preserve too little.

Policy refinement occurs when a policy adds preservation requirements to another. A ring isomorphism policy can be refined to an ordered-ring isomorphism policy by requiring order preservation. A group isomorphism policy can be refined to a topological-group isomorphism policy by requiring homeomorphic compatibility with group operations. A vector-space isomorphism policy can be refined to an inner-product-space isomorphism policy by requiring preservation of the inner product. A categorical equivalence policy can be refined by requiring monoidal, symmetric monoidal, enriched, or higher-categorical structure preservation.

Refinement is crucial because it explains many false substitutions. Suppose X and Y are isomorphic as groups. A model may then assume they are equivalent as topological groups. That move is invalid unless the isomorphism satisfies the refined topological policy. The original certificate does not automatically lift. A certificate for a weaker policy may fail to certify a stronger one. Conversely, a certificate for a stronger policy often induces a certificate for a weaker policy by forgetting structure, but this forgetful passage itself should be recorded.

Policies can also fail. Failure may mean no admissibility certificate exists, or merely that none has been found. Admissibility Algebra distinguishes inadmissible, undecided, and uncertified. Inadmissible means an obstruction has been established. Undecided means the system lacks a proof either way. Uncertified means the model has proposed a claim but no certificate is available. These states must not be collapsed. A mathematical AI should not say false when it means unproved, nor true when it means plausible.

Failure is often witnessed by invariant mismatch. Two groups cannot be isomorphic if their element-order spectra differ. Two topological spaces cannot be homeomorphic if one is compact and the other is not. Two spaces cannot be homotopy equivalent if their homology groups differ. Two rings cannot be isomorphic if one has nilpotent elements and the other does not. A quotient construction fails if the relation is not a congruence with respect to the operations. These obstructions are negative certificates. They are as valuable as positive certificates for training mathematical AI.

The notion of weakest valid policy is especially useful. Given X and Y, the system may report that they are not equal, not isomorphic, but equivalent under a weaker policy. This is not a compromise; it is mathematical precision. Two spaces may not be homeomorphic but may be homotopy equivalent. Two categories may not be isomorphic but may be equivalent. Two formulas may not be syntactically identical but may define the same function on a restricted domain. A good AI should be able to locate the admissibility level at which the claim becomes true.

The notion of strongest available certificate is also useful. If a strong certificate exists, weaker policies may be automatically satisfied by forgetful projection. For example, a homeomorphism yields a homotopy equivalence under standard assumptions. A ring isomorphism yields a group isomorphism of additive groups. A definitional equality may yield a propositional equality. But these implications depend on the surrounding theory and should be encoded as policy maps.

Thus policies form not an unordered list, but a structured system. They have strength relations, refinement maps, forgetful projections, obstruction criteria, and failure states. Making this structure explicit is necessary for proof-grounded AI. Without it, the system may either over-identify objects by using a policy that is too weak or under-identify objects by demanding a policy that is too strong.

4.4 The policy lattice of mathematical identity

The collection of admissibility policies can often be organized as a lattice or partially ordered structure. The ordering is given by preservation strength or implication of admissibility. A policy π_1 is stronger than π_2 if every π_1-admissible identification induces a π_2-admissible identification,

typically by forgetting structure. Equivalently, π_2 is weaker than π_1 if it permits at least the identifications permitted by π_1 and possibly more.

For example, in a suitable setting, equality is stronger than isomorphism, and isomorphism is stronger than weaker structural equivalence. Homeomorphism is stronger than homotopy equivalence. Diffeomorphism is stronger than homeomorphism when smooth structure is present. Isometry is stronger than homeomorphism for metric spaces because it preserves distance, hence also the induced topology. Ordered-field isomorphism is stronger than field isomorphism because it preserves the order relation in addition to field operations.

This ordering is not always linear. Policies can be incomparable. A smooth structure policy and a metric policy may preserve different information. A measure-preserving isomorphism and a topological homeomorphism may overlap but not imply one another without additional compatibility. A monoidal equivalence and an enriched equivalence may preserve different refinements of categorical structure. In such cases the policy system is not a simple hierarchy but a network or lattice-like structure with joins, meets, refinements, and incomparable branches.

The meet of two policies, when it exists, can be interpreted as requiring both preservation conditions. For example, a structure-preserving map that is both topological and algebraic may live under the meet of topological and algebraic policies, producing a topological group policy. A measure-preserving homeomorphism may combine measurable and topological requirements. A monoidal equivalence preserving additional enrichment may combine categorical policies. The meet strengthens identity by requiring simultaneous invariance.

The join of two policies, when meaningful, may represent the coarsest policy weaker than both, or a common abstraction under which both sets of identifications become permitted. This is more delicate. Joining policies may erase more structure than intended. For AI, automatic joins must be handled cautiously because they can create destructive abstractions. A model may be tempted to unify two objects under a broad analogy policy, but such a join may no longer preserve the theorem-relevant invariants.

The policy lattice is valuable because it supports navigation. If a proposed identification fails under π, the system can search downward for weaker policies under which it succeeds. If it succeeds under π, the system can ask whether a stronger certificate exists. If two policies are incomparable, the system can warn that admissibility under one does not imply admissibility under the other. This turns mathematical identity into a computable map rather than a vague judgment.

For a mathematical admissibility database, the policy lattice should be stored explicitly. Policies are nodes. Refinement, implication, forgetful projection, and incompatibility are edges.

Certificates can be transported along some edges. Obstructions can propagate upward or downward depending on the direction of implication. If X and Y fail under a weak policy, they fail under stronger policies that imply it. If X and Y succeed under a strong policy, they succeed under weaker policies reachable by valid forgetful maps. These propagation rules make the database more intelligent than a flat theorem index.

A policy lattice also helps avoid a common AI failure: answering at the wrong level. If asked whether two objects are "the same," the system should not guess the intended policy. It should return a policy-sensitive answer. It might say: equal no, isomorphic no, homotopy equivalent yes, same homology groups yes, homeomorphic unknown. This is not evasive. It is mathematically correct. The system is exposing the lattice of identity rather than flattening it.

In proof assistant contexts, the policy lattice connects to typeclass hierarchies, coercions, equivalences, quotient types, setoids, rewriting systems, and transport mechanisms. Formal systems already encode fragments of policy structure, but they do not necessarily present them as an explicit admissibility lattice. Admissibility Algebra gives this implicit organization a name and a database form.

The lattice metaphor should not be forced where mathematics supplies a more general structure, such as a category, fibration, double category, or higher category of policies and certificates. But as a first approximation, the policy lattice is useful: it teaches the AI that sameness has levels, that levels have implications, and that moving between levels requires permission.

4.5 Same objects, different policies, different judgments

The central practical lesson is that the same pair of objects can yield different judgments under different policies. This is not a pathology. It is ordinary mathematics made explicit. Admissibility Algebra exists because "are X and Y the same?" is rarely a complete mathematical question. The complete question is: same under which policy?

Consider $Z/4Z$ and $Z/2Z \times Z/2Z$. As finite sets, they have the same cardinality. Under the cardinality policy, they are admissibly the same in size. But as groups, they are not isomorphic. $Z/4Z$ has an element of order 4. $Z/2Z \times Z/2Z$ has no element of order 4. The element-order invariant obstructs group-isomorphism admissibility. Thus the pair receives a positive judgment under one policy and a negative judgment under another.

Consider two finite-dimensional vector spaces over the same field with the same dimension. Under the vector-space isomorphism policy, they are structurally the same. But if each vector space has a chosen basis, a distinguished vector, an inner product, a topology, or an operator, additional policies may distinguish them. The bare vector-space certificate does not preserve all

enriched data. A model that says “all vector spaces of the same dimension are the same” must specify the policy and the forgotten structure.

Consider two categories that are equivalent but not isomorphic. Under categorical equivalence, they may be admissibly identified for many categorical purposes. Under strict category isomorphism, the identification may fail. Under equality of formal encodings, it almost certainly fails. A proof assistant may require explicit transport even when a category theorist speaks informally of sameness. The AI must know whether it is reasoning like a human category theorist, a formal kernel, or a database index.

Consider two spaces with the same homology groups. Under a homology-profile policy, they may appear equivalent. Under homotopy equivalence, they may not be. Under homeomorphism, they may certainly not be. Homology is an invariant, not a complete certificate for all stronger policies. AI systems often confuse shared invariants with equivalence. Admissibility Algebra forces the distinction: a shared invariant supports comparison but does not automatically certify identity.

Consider two formal theorem statements, one informal and one Lean-formalized. They may be semantically equivalent under expert interpretation but not yet certified by proof alignment. They may compile but not faithfully represent the intended informal statement. They may prove a nearby theorem with altered hypotheses. The policy of formalization must be explicit: what counts as preserving the theorem? Does it preserve assumptions, quantifiers, constructive content, universe levels, notation, and intended generality? Without that policy, compilation alone is not enough.

These examples show that mathematical AI should return multi-policy judgments. Instead of producing a single answer, it should produce a table of admissibility: policy, result, certificate, obstruction, and substitution scope. The answer “yes” is often incomplete. The answer “no” is often incomplete. The useful answer is: yes under π_1 with certificate C_1; no under π_2 because obstruction O_2; undecided under π_3; equivalent under a weaker quotient π_4; stronger policy π_5 requires additional structure.

This also changes how AI should learn examples. Training data should include contrasts where the same pair appears under different policies. Otherwise the model may memorize a single label. The point is not that $\mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ are “different.” The point is that they are same-size but not same-group. The distinction teaches policy-indexed identity. The model must learn the mapping from policy to invariant to certificate or obstruction.

The same principle applies to future AI beyond mathematics, but mathematics is the clean proving ground. In mathematics, policies can be named, certificates can be formalized, and obstructions can be proved. Therefore mathematical AI can learn the discipline of not collapsing

identity across policies. Once this discipline is learned in formal domains, it can later be generalized to scientific evidence, semantic preservation, memory, tool use, and safety.

The conclusion of Section 4 is direct. Policies must be first-class mathematical objects because equivalence alone does not decide admissibility. A policy determines what kind of sameness is asserted, what invariants must survive, what certificates count, which substitutions are licensed, how failures are recorded, and how identity judgments compare across levels. Mathematical AI cannot be trusted with transformation until it can reason over policies as objects in their own right.

5. Invariants: What Must Not Change

5.1 Invariant preservation as the core of admissibility

An admissibility policy is defined by what it permits to change and what it forbids to change. The forbidden change is the invariant content of the policy. If X is transformed into Y under policy π, the transformation is admissible only if the invariants required by π survive. Thus the heart of Admissibility Algebra is not transformation itself, but invariant-preserving transformation.

An invariant is not merely a property that happens to be unchanged. It is a property that must remain unchanged for the identity judgment to hold. Cardinality is invariant under bijection. Group operation is invariant under group isomorphism. Topological structure is invariant under homeomorphism. Homotopy type is invariant under homotopy equivalence. Truth value is invariant under logical equivalence. Type correctness is invariant under valid formal transformation. Semantic content is invariant under faithful translation. In each case, identity is defined by preservation.

This gives a simple test: to understand a policy, ask what it refuses to lose. A cardinality policy refuses to lose number of elements but may lose order, operation, topology, and meaning. A group-isomorphism policy refuses to lose group structure but may lose element names and presentation. A homeomorphism policy refuses to lose topological structure but may lose metric distances. A homotopy policy refuses to lose deformational structure but may lose pointwise topological detail. A proof-preservation policy refuses to lose derivability under declared assumptions.

The AI relevance is immediate. A mathematical AI that transforms X into Y must know what invariant is being protected. Otherwise it cannot distinguish lawful abstraction from destructive alteration. A summary of a theorem must preserve hypotheses and conclusion. A proof sketch must preserve logical dependency. A formalization must preserve quantifiers, domains,

assumptions, and definitions. A retrieved lemma must preserve applicability conditions. A quotient must preserve operations well enough to remain a quotient in the intended category. A code transformation must preserve behavior under the specified contract.

Current systems often fail because they preserve superficial form while losing invariant content. They keep the vocabulary but change the quantifier. They keep the theorem name but alter the hypotheses. They preserve the shape of an argument but omit the required compactness condition. They preserve semantic proximity but lose formal type. They preserve a diagrammatic resemblance but lose the universal property. These are invariant failures.

Admissibility Algebra therefore treats invariants as explicit components of the policy. A policy π should specify a family $I\pi$ of invariants that must be preserved. A certificate C must show not merely that X and Y are related, but that the relation preserves $I\pi$. An obstruction $O\pi(X,Y)$ should identify an invariant in $I\pi$ that fails to match. This turns identity into an auditable mathematical record.

The slogan is direct: a transformation is admissible only relative to what must not change. Invariant preservation is the content of admissibility.

5.2 Algebraic invariants: order, rank, kernel, spectrum, dimension

Algebra supplies the clearest first examples of invariants. Algebraic objects are defined by operations and relations, and their admissible transformations preserve structural features generated by those operations. When two algebraic objects are compared, invariants provide both positive guidance and negative obstruction.

For finite groups, elementary invariants include order, element-order spectrum, commutativity, center, subgroup structure, normal subgroup lattice, conjugacy classes, derived series, solvability, nilpotency, exponent, automorphism group, and representation data. Some invariants are coarse. Group order alone cannot distinguish $Z/4Z$ from $Z/2Z \times Z/2Z$. Element-order structure can. If one group has an element of order 4 and another does not, no group isomorphism exists. The failed invariant certifies inadmissibility under the group-isomorphism policy.

For rings, invariants include characteristic, units, ideals, prime ideals, maximal ideals, nilpotent elements, idempotents, integral domain status, field status, Noetherianity, Krull dimension, spectrum, module categories, and homological data. A ring isomorphism must preserve much more than cardinality. If one ring has nilpotent elements and another is reduced, they cannot be isomorphic as rings. If one ring is a field and another is not, field status obstructs isomorphism. If spectra differ in ways preserved by ring isomorphism, structural sameness fails.

For modules and vector spaces, invariants include dimension, rank, basis cardinality, torsion, decomposition type, annihilator, endomorphism ring, and module category behavior. Over a field, finite-dimensional vector spaces are classified up to isomorphism by dimension. Over a ring, modules are subtler; rank alone may not classify them. A mathematical AI must not import the vector-space policy into module theory without checking the context Γ. The invariant adequate in one context may be insufficient in another.

For linear operators, matrices, and representations, invariants include rank, determinant, trace, characteristic polynomial, minimal polynomial, eigenvalues, Jordan form where available, singular values, spectrum, kernel, image, nullity, index, and invariant subspaces. Similar matrices share trace, determinant, characteristic polynomial, and spectrum. But sharing eigenvalues may not imply similarity if Jordan structure differs. Thus the invariant set must match the policy. A partial invariant may suggest equivalence but not certify it.

Kernels and images are especially important because they describe loss and preservation under morphisms. A homomorphism with trivial kernel may be injective. A quotient map has kernel encoding the collapsed part. A linear map's kernel records what is destroyed by the transformation. In admissibility terms, the kernel often identifies what the policy permits to be forgotten. For a quotient to be lawful, the kernel or equivalence relation must respect the relevant operations. Otherwise the quotient is destructive.

Algebraic invariants give mathematical AI a natural training ground. They are often computable, formalizable, and connected to explicit obstruction patterns. The AI can learn that two objects may share one invariant but differ in another; that a positive invariant match is not always a complete certificate; that a negative invariant mismatch can be decisive; and that the required invariant depends on the policy. This is exactly the discipline of admissibility.

5.3 Categorical invariants: universal properties, adjunctions, equivalences

Category theory shifts attention from internal element-wise descriptions to relational structure. Categorical invariants are often not properties of isolated objects but properties of objects as determined by their maps, universal roles, functorial behavior, and equivalence classes within a category. This makes category theory a natural home for Admissibility Algebra.

A universal property is an invariant description by admissible mapping behavior. Products, coproducts, pullbacks, pushouts, limits, colimits, equalizers, coequalizers, terminal objects, initial objects, free objects, tensor products, localizations, completions, and representable functors are characterized not primarily by construction but by how they relate to all relevant objects through morphisms. Any two objects satisfying the same universal property are uniquely isomorphic in the appropriate sense. The universal property is therefore a certificate-producing invariant.

This matters for AI because a construction and its universal property are not the same representation. A model may describe a pullback by a diagram, by a set-theoretic subset, by a universal mapping property, by a formal declaration, or by a proof assistant structure. These are admissibly related only if the relevant universal property is preserved. A diagram that looks like a pullback is not a pullback unless the universal condition holds. A generated explanation that names a universal property but fails to preserve its quantification has lost the categorical invariant.

Adjunctions are another central categorical invariant. An adjunction between functors F and G is not merely a pair of maps but a structured relationship between hom-sets or mapping objects, with unit, counit, and coherence. Adjunctions preserve and generate mathematical structure across domains. Free-forgetful relationships, tensor-hom relationships, geometric realization and singular complex, and many algebraic constructions are organized by adjunction. A mathematical AI that recognizes an adjunction-like pattern must still certify the adjunction data. Resemblance of functors is not an adjunction.

Equivalence of categories is itself an admissibility policy with categorical invariants. It preserves the categorical structure relevant to objects and morphisms up to coherent comparison. Properties invariant under equivalence include many categorical constructions and structural features, but not all strict choices. A category equivalent to a small category may be essentially small; a category equivalent to another may share categorical behavior; but chosen representatives, literal object identity, and definitional encodings may differ. The invariant is categorical content, not strict presentation.

Natural transformations and natural isomorphisms supply certificates of coherent comparison. In category theory, pointwise data are often insufficient unless they assemble naturally. AI systems are especially vulnerable here. They may verify a family of object-level correspondences but miss naturality. They may identify components without checking commutative squares. They may treat an object-wise isomorphism as a natural isomorphism. The invariant lost is coherence.

Functoriality is likewise an invariant. A construction that assigns objects to objects but fails to preserve morphisms is not a functor. A transformation that preserves objects but not composition is not admissible under a functorial policy. Mathematical AI must therefore distinguish object-level analogy from functorial structure. Categorical admissibility requires preservation of identities, composition, and natural coherence.

Categorical invariants are powerful because they show that what must not change may be a pattern of relationships rather than a list of internal features. In Admissibility Algebra, this suggests that policies should often be organized fibrationally or categorically: contexts vary,

objects move across contexts, certificates compose, and invariants live over policies. Mathematical AI trained on categorical invariants should learn that sameness is not always element matching. Often it is preservation of a universal role.

5.4 Homotopical invariants: paths, higher paths, deformation classes

Homotopical mathematics treats identity as deformable structure. It asks not merely whether objects are equal or isomorphic, but whether they can be continuously transformed into one another while preserving the relevant shape. This provides a deeper training ground for policy-indexed identity because homotopy theory contains many levels of sameness.

At the most familiar level, homotopy equivalence identifies spaces that have the same homotopy type. Two spaces X and Y are homotopy equivalent if there exist maps $f : X \to Y$ and $g : Y \to X$ such that $g \circ f$ is homotopic to id_X and $f \circ g$ is homotopic to id_Y. The certificate is not merely a pair of maps but maps together with homotopies. The invariant is deformational structure rather than exact point-set topology.

Homotopical invariants include fundamental group, higher homotopy groups, homology, cohomology, cohomology rings, characteristic classes, stable homotopy data, and other structures depending on the context. Some invariants are complete in special settings; many are not complete in general. A mathematical AI must learn that shared homology is evidence of similarity but not a universal certificate of homotopy equivalence. Conversely, a mismatch in homology can obstruct homotopy equivalence.

Paths and higher paths make identity itself structured. In homotopy type theory, an equality proof can be interpreted as a path, and paths can have paths between them. Identity is no longer a mere proposition but may carry higher-dimensional structure. This is deeply aligned with Admissibility Algebra because it refuses to treat identity as an unexamined primitive. It asks what witnesses identity, how witnesses compose, and how coherence among witnesses is maintained.

Univalence sharpens this further. In homotopy type theory, equivalence of types can be treated as identity under the univalence principle. But even here, admissibility matters. The policy is not vague equivalence; it is equivalence of types within a specified type-theoretic universe, with transport governed by the formal rules of the system. A mathematical AI using HoTT-like reasoning must know when equivalence can be transported as identity and what data are required.

Homotopical thinking also clarifies quotienting. Homotopy quotients, higher inductive types, stacks, orbifolds, and moduli problems show that quotienting by an equivalence relation may not be enough. Stabilizers, automorphisms, paths, and higher coherence may matter. A naive quotient may collapse too much. A homotopy quotient preserves information about how points

are identified, not merely that they are identified. This is an important warning for AI abstraction: collapsing representatives into equivalence classes may erase witness structure required for later reasoning.

Deformation classes also illustrate policy strength. Homeomorphism preserves more than homotopy equivalence. Homotopy equivalence preserves more than homology equivalence. Weak homotopy equivalence has its own jurisdiction. Stable equivalence may be weaker still. Each policy permits different identifications and preserves different invariants. A mathematical AI must know which homotopical policy is active before declaring objects "the same."

Homotopy theory therefore supplies one of the most sophisticated examples of admissibility. It shows that identity can be witnessed, layered, composed, weakened, strengthened, and coherently transported. For mathematical AI, this matters not only in topology but in all domains where equivalence is structured. A model must not merely recognize that two objects are "equivalent." It must identify the level of equivalence, the path or higher-path data, the invariants preserved, and the substitutions licensed.

5.5 AI failure as invariant loss

Many failures of mathematical AI can be reclassified as invariant loss. The system transforms X into Y, but Y no longer preserves the invariant required by the governing policy π. This reframing is useful because it converts vague criticism into a mathematical diagnosis.

A theorem summary fails when it loses a hypothesis, changes a quantifier, strengthens a conclusion, weakens an assumption, changes the domain, or suppresses an exception. The invariant that should have been preserved is the theorem's logical content relative to context Γ. A summary may be shorter, clearer, and plausible while still inadmissible if it changes the formal dependency structure.

A proof sketch fails when it preserves rhetorical sequence but loses logical necessity. It may say "by compactness" where compactness is not available, "by continuity" where uniform continuity is needed, or "by induction" without a valid induction parameter. The invariant lost is derivability. The proof-shaped text is not a proof certificate.

A formalization fails when it compiles but represents the wrong theorem. It may replace a dependent condition with a simpler one, alter universe levels, assume decidability, use classical choice where constructive content was intended, or change a structure by choosing a different definition. Compilation certifies syntactic acceptance by the formal system, not necessarily semantic fidelity to the informal source. The missing invariant is faithful correspondence.

A retrieval system fails when it retrieves a nearby but inapplicable theorem. Similar language, overlapping symbols, or shared topic embeddings do not guarantee admissibility. The retrieved

theorem may require finite-dimensionality, commutativity, compactness, smoothness, normality, completeness, or a stronger morphism than the current context provides. The invariant lost is applicability under Γ.

A quotienting system fails when it collapses distinctions that the policy should preserve. It may identify objects by shared cardinality when structure matters, by shared homology when homotopy type matters, by semantic similarity when proof identity matters, or by notation when formal type matters. The invariant lost is the structure that made the original distinction meaningful.

A theorem-proving model fails when it treats a tactic pattern as transferable without preserving the proof obligations. Tactic scripts are not magic incantations. They work because the proof state satisfies certain structural conditions. If those conditions are absent, the same script may fail or prove a different goal. The invariant lost is proof-state compatibility.

This diagnosis suggests a training strategy. Instead of training only on correct answers, train on invariant-preservation and invariant-loss examples. Show the model pairs that look similar but fail under specific policies. Teach it to identify which invariant broke. Teach it to distinguish a decisive obstruction from an incomplete search. Teach it that preserving surface form is not enough.

For example, a training record should not merely say that $\mathbb{Z}/4\mathbb{Z}$ is not isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. It should say: under cardinality policy, admissible; under group-isomorphism policy, inadmissible; obstruction: element-order invariant; certificate of obstruction: one has an element of order 4 and the other does not. This teaches the model the full admissibility pattern.

Likewise, a record comparing two spaces should distinguish homeomorphism, homotopy equivalence, homology equivalence, and weak equivalence. A record comparing two theorem statements should identify which hypotheses are preserved and which are lost. A record comparing two formalizations should specify whether semantic fidelity, proof acceptance, or definitional equality is being judged.

Invariant loss is the mathematical form of hallucination. The model may not intend to deceive; it may simply fail to track what must not change. Admissibility Algebra supplies the corrective: every transformation must declare its policy, its invariants, and its certificate. If the required invariant is lost, the transformation is inadmissible.

The conclusion of Section 5 is therefore straightforward. Invariants are the content of policy-indexed identity. Policies say what counts as the same; invariants say what must survive for that sameness to be lawful. Mathematical AI becomes trustworthy only when it can detect, preserve, certify, and learn from invariants. Without invariants, there is only transformation. With invariants, transformation becomes admissible.

6. Certificates and Obstructions

6.1 A certificate is not an explanation but a witness

A certificate is the object, construction, proof, computation, map, homotopy, reduction, derivation, or formal trace that witnesses admissibility. It is not merely an explanation. An explanation may help a human understand why X and Y should count as the same under policy π, but a certificate is what authorizes the judgment $\Gamma \vdash X \equiv_\pi Y : C$. The explanation belongs to pedagogy. The certificate belongs to admissibility.

This distinction is crucial for mathematical AI. Large models are good at explanations, especially when explanations are judged by fluency, plausibility, and local coherence. They can say why two structures "should be equivalent," why a proof step "follows," why a quotient "makes sense," or why a theorem "applies." But unless the explanation contains or points to the actual witness required by the policy, it does not certify admissibility. A persuasive explanation may still conceal an invalid substitution.

For equality, the certificate may be a proof of equality. For definitional equality, it may be a reduction recognized by the formal kernel. For isomorphism, it may be a structure-preserving inverse pair. For homotopy equivalence, it may be maps plus homotopies. For categorical equivalence, it may be functors plus natural isomorphisms or full-faithful-essentially-surjective data. For quotient equality, it may be evidence that two representatives lie in the same equivalence class. Each policy determines what kind of witness counts.

A certificate may be compact or large, explicit or implicit, constructive or nonconstructive, formal or informal, canonical or noncanonical. These distinctions matter. A nonconstructive existence proof may certify that an isomorphism exists, but it may not provide the computational data required to transport a construction. A canonical certificate may allow stable database indexing; a noncanonical certificate may require additional choices. A proof assistant certificate may be machine-checkable, while an informal certificate may require human interpretation. An admissibility system must record these differences rather than treating all justifications as the same.

The certificate is also what prevents policy-indexed identity from becoming arbitrary. Without C, the statement $X \equiv_\pi Y$ is a claim. With C, it becomes a certified admissibility judgment. The certificate ties the assertion of sameness to a checkable mathematical object. In proof-grounded AI, this is the difference between generation and verification. The AI may generate the candidate Y, but the admissibility layer must locate, construct, or reject the certificate C.

Explanations remain valuable, but they should be downstream from certificates. The ideal mathematical AI should produce both: a certificate for the machine or expert system, and an explanation for the human reader. If the certificate cannot be produced, the explanation must

not pretend that the judgment is established. The system may say that the transformation is plausible, conjectural, analogous, or suggested by evidence, but it should not present it as admissible.

This is the first law of certificates: an explanation may describe admissibility, but only a witness certifies it.

6.2 Proof objects, maps, equivalences, homotopies, and computations

Certificates take different mathematical forms because policies take different mathematical forms. There is no single universal certificate. Admissibility Algebra therefore treats certificates as indexed families: for each policy π and objects X,Y in context Γ, there is a certificate type or certificate class $C\pi(X,Y)$. To assert $X \equiv\pi Y$ is to assert that this certificate class is inhabited by a valid witness.

In proof assistants, proof objects are the clearest certificates. A theorem is not certified by a persuasive paragraph but by a term inhabiting the theorem's type, accepted by the kernel. The proof object may be generated directly, elaborated from tactics, or reconstructed from intermediate proof scripts. Once accepted, it certifies the proposition relative to the formal system, definitions, imports, universe levels, and assumptions. In this setting, the certificate is rigorous because the admissibility conditions are enforced mechanically.

In algebra, maps often serve as certificates. A group isomorphism is certified by a map preserving multiplication, together with invertibility or a structure-preserving inverse. A module isomorphism is certified by a linear map with inverse. A ring isomorphism is certified by preservation of addition, multiplication, and units according to the policy. A field isomorphism must preserve field operations. These maps are not merely correspondences. They are admissibility witnesses because they preserve the operations defining the structure.

In category theory, certificates are often higher-structured. An equivalence of categories may be certified by functors $F : C \to D$ and $G : D \to C$, together with natural isomorphisms $IdC \cong G \circ F$ and $F \circ G \cong IdD$. Alternatively, it may be certified by a functor that is full, faithful, and essentially surjective, depending on the formal setting. Natural transformations, adjunction units and counits, commutative diagrams, and universal property witnesses may all serve as certificates. Here, admissibility depends not only on object-level correspondence but on functorial and natural coherence.

In homotopy theory, certificates include paths, homotopies, equivalences, and higher coherence data. A homotopy equivalence requires maps in both directions and homotopies from composites to identities. In homotopy type theory, an equivalence of types may serve as a certificate that can be transported into identity by univalence. But the certificate is not a vague

assertion that the types “have the same structure.” It is a structured object satisfying the rules of equivalence in the type-theoretic universe.

In computation, certificates may be reductions, normal forms, execution traces, symbolic derivations, decision procedure outputs, test suites, formal verification results, or proof-carrying code. If a symbolic expression is simplified, the certificate may be a chain of rewrite rules. If an algorithm is replaced by an optimized algorithm, the certificate may be a proof of behavioral equivalence. If an approximate computation is accepted, the certificate may include error bounds. If a theorem is decided by computation, the certificate may include a verified computation trace.

These examples show that certificate type is part of policy design. A policy that cannot specify what counts as a certificate is incomplete. The mathematical AI may still generate plausible claims, but it cannot certify them. Therefore the admissibility database must not store only that X and Y are related. It must store the certificate kind: proof object, explicit map, equivalence data, homotopy data, computation trace, invariant comparison, quotient relation membership, or obstruction.

Certificates also have internal validity conditions. A proposed group isomorphism must be checked. A proposed natural transformation must satisfy naturality. A proposed homotopy must have the right endpoints. A proposed proof term must type-check. A proposed quotient certificate must respect the equivalence relation. A proposed computation certificate must be reproducible or formally verified. Thus the certificate is not merely data; it is data plus validation.

For mathematical AI, the ideal output is therefore not only a theorem-like sentence, but a certificate-bearing record. The model should say what certificate type is required, whether it has been found, whether it has been checked, and what substitutions it licenses. This transforms mathematical AI from a fluent proposer into a proof-grounded admissibility assistant.

6.3 Obstructions as certified non-admissibility

A negative result can also be certified. If no admissibility certificate exists because a required invariant fails, the obstruction is itself a witness of non-admissibility. Obstructions are the negative counterpart of certificates. They show that $\Gamma \nvdash X \equiv_{\pi} Y$ because some condition required by π cannot be satisfied.

An obstruction is stronger than failure to find a proof. A model may fail to construct an isomorphism because the search was inadequate. That is not yet a proof that no isomorphism exists. But if the model identifies an invariant preserved by all isomorphisms and shows that X

and Y differ in that invariant, then the non-isomorphism is certified. The obstruction establishes the boundary of admissibility.

For example, $\mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ are not isomorphic as groups because element-order structure differs. Any group isomorphism preserves orders of elements. $\mathbb{Z}/4\mathbb{Z}$ contains an element of order 4. $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ does not. This is not merely an unsuccessful search for an isomorphism. It is a certificate that no group-isomorphism certificate can exist.

Similarly, two topological spaces cannot be homeomorphic if one is compact and the other is not, assuming compactness is preserved in the relevant context. Two spaces cannot be homotopy equivalent if their homology groups differ. Two rings cannot be isomorphic if their characteristics differ. Two fields cannot be isomorphic if one is algebraically closed and the other is not. Two matrices cannot be similar if their characteristic polynomials differ, though matching characteristic polynomials may not be sufficient. Each obstruction identifies an invariant required by the policy.

Obstructions are especially useful for mathematical AI because they prevent endless generation. A system without obstruction knowledge may repeatedly search for a certificate that cannot exist. A system with obstructions can stop, explain failure, and suggest a weaker or different policy. It can say: not isomorphic as groups, but same cardinality; not homeomorphic, but possibly homotopy equivalent; not definitionally equal, but propositionally equal; not equal as ordered fields, but isomorphic as fields after forgetting order.

Obstructions also protect against destructive abstraction. A quotient construction may fail because the proposed equivalence relation is not compatible with the operations. This obstruction tells the system that the collapse would not produce a well-defined algebraic object. For example, quotienting a group-like structure by an arbitrary subset rather than a normal subgroup generally fails to preserve group structure. The obstruction is not philosophical; it is operational. The multiplication of equivalence classes would be ill-defined.

In formal systems, obstructions may appear as type errors, failed unification, missing instances, failed coercions, unsatisfied hypotheses, universe inconsistencies, or kernel rejection. These failures should not be treated as mere debugging noise. Many encode admissibility boundaries. A failed rewrite may reveal that equality is not available at the required type. A failed theorem application may reveal that a hypothesis is missing. A failed coercion may reveal that two structures have not been connected by an admissible map.

For an admissibility database, obstructions should be stored with as much care as positive certificates. A record should identify the policy, the failed judgment, the violated invariant, the proof of violation, and the strongest conclusion justified by the obstruction. This teaches the AI not only what transformations work, but why some transformations are forbidden.

The general principle is: negative admissibility is not absence; it is structured refusal. A mature mathematical AI should know how to say no with a certificate.

6.4 Counterexamples, invariant mismatches, and failed transports

Obstructions appear in several recurring forms. The most important are counterexamples, invariant mismatches, and failed transports. Each diagnoses a different kind of inadmissibility.

A counterexample refutes a proposed universal admissibility claim. If an AI asserts that every bijective continuous map is a homeomorphism, a counterexample shows that the proposed transformation policy is invalid without additional hypotheses. If an AI asserts that equal cardinality implies group isomorphism, $Z/4Z$ and $Z/2Z \times Z/2Z$ refute the claim. If an AI asserts that shared homology groups imply homotopy equivalence, known counterexamples refute the policy. Counterexamples are essential because they correct overgeneralized policies.

Invariant mismatch is the most direct obstruction. A policy π requires preservation of invariants $I\pi$. If X and Y differ in an invariant in $I\pi$, then $X \equiv_\pi Y$ is impossible. The invariant may be cardinality, dimension, rank, determinant, trace, group order, element-order spectrum, compactness, connectedness, characteristic, homology, spectrum, type, proof obligation, or categorical universal property. The mismatch says: the proposed sameness forgets something that π forbids forgetting.

Failed transport is subtler. It occurs when a system tries to move a theorem, construction, proof, or property across an asserted equivalence, but the policy does not license that movement. Transport requires more than a relation. It requires an admissible route from X to Y preserving the property being transported. An isomorphism of groups may transport group-theoretic properties, but not properties of a particular presentation unless the presentation data are included. A homotopy equivalence may transport homotopy-invariant properties, but not arbitrary topological properties. An equivalence of categories may transport categorical structure, but not strict object equality.

Failed transport is a common mathematical AI error because it often occurs after a partially valid identification. The system may correctly recognize that X and Y are equivalent under some policy, but then use the equivalence in a context requiring a stronger policy. The first judgment is true; the substitution is false. This shows why admissibility must include substitution scope. A certificate does not license every future use. It licenses only the uses compatible with π.

A formal proof assistant exposes failed transport through type errors, missing coercions, or inability to rewrite. Informal mathematical AI may conceal the same error in prose. It may say "since these are equivalent, the result follows," while the required property is not invariant under that equivalence. The admissibility layer must ask: what property is being transported, and is it invariant under π?

Counterexamples, invariant mismatches, and failed transports should be explicitly represented in training data. The system should learn that an obstruction is not a generic failure. It has a type. A counterexample refutes a proposed rule. An invariant mismatch blocks an identification. A failed transport blocks a substitution across an otherwise valid relation. These distinctions help the AI produce more precise diagnoses.

For example, suppose a model compares two spaces. If one is compact and the other is noncompact, homeomorphism is obstructed by compactness mismatch. If their homology groups differ, homotopy equivalence is obstructed. If their homology groups agree but no homotopy equivalence has been established, the correct status is not admissible but undecided or insufficiently certified. If a theorem about metric completeness is transported across homeomorphism, transport may fail because completeness is not purely topological. These are different cases.

The failed-transport problem also motivates policy refinement. If transport fails because π is too weak, the system may seek a stronger policy π'. For example, a metric property may require isometry rather than homeomorphism. A smooth property may require diffeomorphism rather than homeomorphism. An ordered property may require ordered-ring isomorphism rather than ring isomorphism. The AI should be trained to identify the policy strengthening needed to make transport admissible.

Thus obstructions are not merely rejections. They are guides to the correct policy. They tell the system what must be added, changed, weakened, or strengthened to make a transformation lawful.

6.5 The admissibility record: admissible, inadmissible, undecided

A mathematical AI should not collapse all judgments into true and false. Admissibility requires at least three primary states: admissible, inadmissible, and undecided. A fourth practical state, uncertified, is also useful. These states correspond to different epistemic and mathematical conditions.

Admissible means that a valid certificate C has been found, checked, and recorded under policy π in context Γ. The system may then assert $\Gamma \vdash X \equiv_\pi Y : C$ and specify the substitution scope. The output may be used downstream according to the policy. If the certificate is formal, the record should point to the proof object, theorem, construction, or kernel-checked artifact. If informal, the record should state the certificate type and verification status.

Inadmissible means that an obstruction O has been found and certified. The system may assert $\Gamma \nvdash X \equiv_\pi Y$ because $O_\pi(X,Y)$. The obstruction should identify which invariant fails, which hypothesis is missing, which transport is invalid, or which counterexample applies. Inadmissibility is stronger than lack of proof. It is a negative result.

Undecided means that neither a certificate nor an obstruction is currently available. The system has not established admissibility, but it has also not established failure. This state is essential in mathematics, where search may be incomplete and many questions are difficult or open. A trustworthy AI must be comfortable saying undecided. It should not convert undecided into plausible yes or cautious no merely to satisfy conversational expectations.

Uncertified means that the system or user has proposed a claim, and the claim may be plausible or even likely, but no certificate has been attached. This is the natural state of many generated suggestions. A theorem prover may propose a lemma; an LLM may propose an equivalence; a retrieval system may suggest a related theorem. Until certification occurs, the claim remains uncertified. It may enter a search queue, but it should not enter the admissibility database as established.

These states should be represented in a standard admissibility record. The record should include Γ, X, Y, π, result status, certificate C if positive, obstruction O if negative, confidence or search metadata if undecided, source references, formal-system information, verification status, and allowed substitution scope. This record is the atomic data structure of mathematical admissibility AI.

A mature system should also track certificate strength. A fully formal Lean proof has a different status from an informal proof sketch. A verified computation has a different status from an empirical test. An explicit isomorphism has a different status from an existence proof. A canonical equivalence has a different status from a choice-dependent equivalence. The admissibility state should therefore include not only positive or negative result, but certificate quality.

Likewise, the system should track policy alternatives. If X and Y are inadmissible under π, they may be admissible under a weaker policy π'. If they are admissible under π, a stronger policy π'' may remain undecided. The record should therefore support multi-policy comparison. This is particularly important for AI explanations. The user should learn not merely whether X and Y are the same, but at what level of sameness the claim holds.

The admissibility record also supports learning. Training examples can include positive certificates, negative obstructions, undecided cases, and uncertified proposals. This teaches the AI epistemic discipline. It learns that mathematical reasoning includes proof, disproof, open status, and speculation. It learns that plausibility is not admissibility. It learns that failure to find a proof is not proof of failure. It learns that a claim can be correct under one policy and false under another.

The conclusion of Section 6 is that certificates and obstructions are the two pillars of proof-grounded admissibility. Certificates authorize transformation. Obstructions prohibit it.

Undecided and uncertified states prevent false closure. A mathematical AI organized around these records will not merely produce mathematical text; it will maintain an auditable boundary between what has been certified, what has been refuted, what remains open, and what has only been guessed.

7. Scoped Substitution

7.1 The failure of global replacement rules

Substitution is one of the most powerful operations in mathematics, but it is also one of the most dangerous. To substitute Y for X is to act as if Y can stand in the place of X without damaging the surrounding argument, construction, computation, proof, or interpretation. Elementary algebra teaches substitution as a routine move: equal expressions may replace one another. But outside rigid equality, substitution becomes policy-dependent. The fact that X and Y are related does not imply that Y may globally replace X.

A global replacement rule says, in effect, that once X ≡ Y has been established, every occurrence of X may be replaced by Y. This is valid only under very strong identity policies and only when the surrounding context respects that identity. In formal systems, even equality may require care when dependent types, coercions, definitional unfolding, universe levels, or proof irrelevance are involved. In ordinary mathematical practice, replacement is even more delicate. Isomorphic groups may be interchangeable for group-theoretic properties, but not for properties depending on presentation, embedding, labeling, topology, computation, or chosen generators. Homotopy equivalent spaces may be interchangeable for homotopy-invariant properties, but not for arbitrary topological or metric properties. Equivalent categories may be interchangeable for categorical purposes, but not for strict object equality or implementation-specific definitions.

Thus the failure of global replacement is not an edge case. It is the normal condition of structural mathematics. Most mathematical sameness is scoped. It applies only to contexts compatible with the policy that certified the sameness. An isomorphism does not erase every distinction. A quotient does not preserve everything. An equivalence does not authorize all transport. A semantic similarity does not license theorem application. A retrieved lemma does not automatically fit the proof state.

This is one of the central weaknesses of mathematical AI. A model may correctly recognize a relation between X and Y but incorrectly use the relation outside its substitution scope. It may know that two objects are “equivalent” but fail to know which properties are invariant under that equivalence. It may replace a theorem by a nearby theorem whose hypotheses are different. It may formalize an informal statement in a way that preserves the conclusion but

changes the domain. It may summarize a proof in a way that preserves the narrative but loses the dependency structure. These are all failures of scoped substitution.

The remedy is to attach substitution rules to policies. A policy π does not merely say when X and Y count as the same. It also says in which contexts that sameness licenses replacement. The judgment $\Gamma \vdash X \equiv_\pi Y : C$ should therefore carry a substitution scope S_π. This scope specifies the formulas, constructions, invariants, proof obligations, or contexts in which Y may replace X. Without S_π, an admissibility certificate is incomplete for AI use.

Scoped substitution also prevents overcorrection. The conclusion is not that AI must avoid replacement unless literal equality holds. That would destroy the power of modern mathematics. The conclusion is that replacement must be indexed by the policy that authorizes it. A good mathematical AI should freely use isomorphism, equivalence, homotopy, quotienting, and transport, but only within their lawful scope. The correct standard is not rigid literalism; it is disciplined permission.

7.2 Substitution only under policy

The basic rule of Admissibility Algebra is that substitution is allowed only under a policy. If X is to be replaced by Y, the system must know which π licenses the replacement and which invariants π preserves. The expression $X \equiv_\pi Y$ is therefore not merely a statement of sameness. It is a conditional permission to substitute Y for X in π-compatible contexts.

This rule can be stated informally:

If $\Gamma \vdash X \equiv_\pi Y : C$, then Y may replace X only in contexts that respect π.

The context of replacement is as important as the certificate itself. A group isomorphism permits transport of group-theoretic statements. It does not automatically permit transport of statements about a specific Cayley table encoding, a named generator, a chosen embedding, a topology, or a computational representation. A homeomorphism permits transport of topological properties. It does not automatically preserve metric distance, smoothness, measure, or geodesic structure. A homotopy equivalence preserves homotopy-invariant data. It does not preserve all point-set properties. The policy determines the legal zone of substitution.

In proof assistants, this idea appears in rewrite rules, congruence lemmas, transport functions, coercions, equivalences, quotient recursors, and typeclass-controlled instances. A rewrite is valid only when the equality has the correct type and occurs in an appropriate syntactic position. Transport across equality may be necessary when types depend on the object being replaced. Substitution in dependent type theory is not mere textual replacement; it is structured transport along an identity witness. This gives a formal model for scoped substitution.

In informal mathematics, the same discipline is present but often hidden. When mathematicians work “up to isomorphism,” they implicitly restrict attention to isomorphism-invariant properties. When they work “up to homotopy,” they restrict attention to homotopy-invariant properties. When they work “modulo gauge,” they restrict attention to gauge-invariant observables. When they classify objects “up to equivalence,” they choose a quotient world in which certain distinctions are intentionally ignored. These phrases are policy declarations.

For mathematical AI, every such phrase should become machine-readable. The model should not simply say “we can replace X by Y.” It should say “under π, replacement is allowed for properties in Sπ.” This requirement is especially important for retrieval-augmented systems. A retrieved theorem may be applicable only if the current goal lies within the substitution scope of the theorem’s policy. If not, the retrieval is merely suggestive.

The policy requirement also applies to generated definitions. If an AI introduces a new definition claimed to be equivalent to an old one, the system must specify the policy: definitionally equal, propositionally equivalent, isomorphic, naturally isomorphic, equivalent in a category, equivalent under a quotient, or merely extensionally equal. Each policy gives different substitution rights. A definitionally equal replacement may reduce computationally. A propositionally equivalent replacement may require explicit rewrite. An isomorphic replacement may require transport. A merely analogous replacement may license no substitution at all.

Thus substitution only under policy is the operational version of admissibility. Identity is not a global pass. It is a permission indexed by π, certified by C, and limited by Sπ. Mathematical AI must learn to ask not only whether X and Y are related, but what that relation allows it to do.

7.3 Transport across equivalence and its limits

Transport is the disciplined movement of structure, properties, proofs, or constructions across an admissible identification. If X and Y are related by an appropriate certificate C under policy π, then some information about X may be carried to Y. But transport is not automatic. What can be transported depends on the policy and on the invariance of the transported object.

In ordinary equality, substitution and transport often appear trivial. If $x = y$, then a property $P(x)$ can usually be transported to $P(y)$. In dependent type theory, this transport is explicit: a proof of equality induces a way to move terms across dependent families. Even here, the mechanism is not mere textual replacement. It depends on the identity witness and the type family over which transport occurs. This explicitness is valuable because it shows what informal mathematics often suppresses.

Across isomorphism, transport is less rigid but still structured. A group isomorphism allows transport of group-theoretic properties. If G is abelian and $G \cong H$, then H is abelian. If G has a

subgroup of a certain structural kind, an isomorphism transports it to a corresponding subgroup of H. But not every property of G is transported. If G is represented as a particular permutation group, the property of containing a named permutation does not transport unless the representation and naming data are included. The isomorphism transports structure, not incidental presentation.

Across categorical equivalence, transport becomes more subtle. Properties invariant under equivalence may be transported, but strict equality-sensitive data may not. Equivalence preserves categorical content up to coherent comparison, but it does not identify objects on the nose. A theorem stated in equivalence-invariant form may transport; a theorem depending on chosen representatives may require additional data. Natural coherence becomes part of the certificate. Without it, transport may fail even if object-wise similarities exist.

Across homotopy equivalence, transport is scoped to homotopy-invariant properties. Fundamental group, homology, cohomology, and homotopy groups may transport under suitable conditions. But local topological details, metric properties, smooth structures, and exact embeddings generally do not. A mathematical AI must therefore know whether the property being moved is homotopy-invariant. If it is not, the transport is inadmissible.

The limits of transport are a major source of AI error. A model may observe that X and Y are equivalent in some sense and then transfer a property not preserved by that sense. For example, it may infer homeomorphic from homotopy equivalent, isometric from homeomorphic, equal from isomorphic, or strict equality from categorical equivalence. These are transport failures. They arise because the system treats equivalence as universal substitutability.

Admissibility Algebra handles this by making transport a derived operation, not a primitive entitlement. The system must possess $\Gamma \vdash X \equiv_\pi Y : C$ and must verify that the property, theorem, construction, or proof being transported lies within S_π. If P is π-invariant, then P(X) may be transported to P(Y). If P is not π-invariant, no transport is licensed. If invariance is unknown, the transport is undecided.

This suggests a database structure. For each policy π, store known invariant predicates, transportable constructions, nontransportable properties, and examples of failed transport. Then the AI can query whether a proposed movement is admissible. It can say: “This property is invariant under isomorphism but not under homotopy equivalence,” or “This theorem requires a stronger policy than the one currently certified,” or “Transport is possible after enriching the certificate with topology.”

Transport is the mechanism by which admissibility becomes useful. Without transport, equivalence is inert. With uncontrolled transport, equivalence becomes dangerous. The correct middle is scoped, certificate-bearing transport across policy-indexed identity.

7.4 Context-sensitive rewriting in theorem proving

Rewriting is the operational form of substitution in theorem proving. A theorem prover rewrites expressions using equalities, equivalences, simplification lemmas, reduction rules, congruence rules, and definitional unfoldings. Rewriting is powerful because it can transform a difficult goal into an easier one without changing its truth under the relevant formal policy. But rewriting is valid only when the rule applies in the current context and preserves the proof obligation.

In formal systems, rewriting is context-sensitive by necessity. A rewrite rule has a type. It may require hypotheses. It may apply only after variables are instantiated. It may require a matching subterm. It may interact with definitional equality, coercions, typeclass inference, universe levels, or dependent types. A rewrite that appears obvious informally may fail formally because the required equality is not of the right form. Conversely, a formal simplifier may perform many valid rewrites that would be tedious to display by hand. In all cases, rewriting is governed by admissibility conditions.

The difference between definitional equality and propositional equality is especially instructive. Definitional equality allows the kernel to identify terms by computation or unfolding. Propositional equality requires a proof. Both can support replacement, but through different mechanisms. A model that confuses them may produce formal code that seems mathematically reasonable but fails to elaborate. The policy of equality matters at the proof-engine level.

Rewriting under equivalence rather than equality is more delicate. In some contexts, propositions may be rewritten by logical equivalence; structures may be transported along isomorphism; quotient representatives may be replaced by related representatives; setoid rewriting may permit replacement under a declared equivalence relation. Each case requires additional machinery. The rewrite is not licensed merely because two expressions are “related.” The relation must be recognized as a congruence for the surrounding context.

Congruence is central. A relation R supports substitution through a function f only if x R y implies f(x) R f(y) under the relevant policy. If this compatibility fails, rewriting inside f is inadmissible. This is why quotient construction requires relations compatible with operations. If addition is not well-defined on equivalence classes, one cannot rewrite representatives freely. Context-sensitive rewriting is therefore the local proof-theoretic face of quotient admissibility.

Mathematical AI must learn this discipline because many generated proof steps are failed rewrites. The model sees a familiar transformation and applies it where the side conditions do not hold. It cancels a factor that may be zero. It divides by an expression without proving

nonzero. It rewrites under a binder without respecting variable dependence. It applies a theorem after changing the domain. It simplifies a quotient expression without proving the relation is compatible. These are not random mistakes; they are context-sensitive rewriting failures.

A proof-grounded AI should therefore represent rewrite rules as admissibility records. Each rule should specify the source pattern, target pattern, policy, required hypotheses, invariant preserved, context restrictions, and certificate. A successful rewrite produces a trace. A failed rewrite produces an obstruction: missing hypothesis, type mismatch, non-congruence, failed unification, or policy mismatch. These records become training data for admissible theorem proving.

Context-sensitive rewriting also provides a bridge from formal proof assistants to informal mathematical writing. When a human says “simplifying,” “identifying,” “without loss of generality,” “passing to the quotient,” “transporting along the equivalence,” or “replacing by an isomorphic object,” each phrase hides a rewriting policy. A mathematical AI should expose the hidden rule. It should ask what is being rewritten, under what equivalence, with what side conditions, and in which context.

Thus theorem proving teaches the operational necessity of scoped substitution. Rewriting is not text editing. It is certified transformation inside a context. An admissibility-aware AI must treat every rewrite as a judgment, not a habit.

7.5 The rule: equivalence without admissibility does not license replacement

The governing rule of this section is simple: equivalence without admissibility does not license replacement. A relation between X and Y may be interesting, suggestive, useful for search, or even mathematically important, but it does not authorize substitution unless a policy π specifies that the relation preserves the invariants required by the target context.

This rule prevents the most common overreach in mathematical AI. The system finds that two objects are similar and replaces one by the other. It finds that two theorems have related statements and applies one as if it were the other. It finds that two categories are equivalent and treats them as strictly equal. It finds that two spaces have the same homology and treats them as homotopy equivalent. It finds that an informal proof resembles a known argument and treats the proof as complete. In each case, some relation exists, but the replacement is not licensed.

Admissibility requires more than relation. It requires policy, invariant preservation, certificate, and scope. The full replacement rule is:

Given $\Gamma \vdash X \equiv_\pi Y : C$, replacement of X by Y is permitted only in contexts whose required invariants are preserved by π.

If the context demands a stronger policy than π, replacement fails. If the property being transported is not π-invariant, replacement fails. If no certificate C is available, replacement is uncertified. If an obstruction exists, replacement is inadmissible. If the policy is unclear, the system should expose the ambiguity rather than silently proceed.

This rule is not a limitation on creativity. It separates search from certification. During search, a mathematical AI may use analogies, similarities, embeddings, partial invariants, informal heuristics, and weak correspondences. These can guide conjecture and retrieval. But before a result is asserted, used in proof, stored as knowledge, or offered as certified output, the transformation must pass through admissibility. Heuristic relation may propose; admissibility must dispose.

The rule also allows nuanced answers. A system may say: “X may replace Y for cardinality arguments but not for group-theoretic arguments.” Or: “This equivalence permits homotopy-invariant conclusions but not metric conclusions.” Or: “The retrieved theorem is relevant but cannot be applied because compactness is missing.” Or: “The formalization compiles, but semantic equivalence to the informal theorem is not certified.” These are the kinds of answers that distinguish proof-grounded AI from fluent generation.

Scoped substitution should therefore be part of the core training curriculum. The model should be given examples where replacement is allowed under one policy and forbidden under another. It should learn side conditions, invariant classes, failed transport, and quotient compatibility. It should learn to ask which context the replacement is being used in. It should learn to refuse unauthorized substitution even when the result sounds mathematically elegant.

The public version of the rule is: “same” is not enough. Same under what policy, for what use, preserving what invariant, with what certificate? This question is the operational heart of Admissibility Algebra.

The conclusion is that mathematical AI becomes trustworthy when every substitution is scoped. Equality, isomorphism, equivalence, homotopy, quotienting, and analogy all have roles, but none has unlimited jurisdiction. A future mathematical AI must not merely know that X and Y are related. It must know what that relation permits.

8. Quotients as Controlled Collapse

8.1 Quotienting as permitted forgetting

A quotient is a disciplined act of forgetting. It identifies objects, elements, states, paths, proofs, or presentations according to a rule and then treats the resulting equivalence classes as the new objects of study. This is one of the most powerful moves in mathematics. It is also one of the most dangerous. A quotient can reveal structure by removing irrelevant distinctions, or it can destroy structure by collapsing distinctions that the policy should have preserved.

In elementary settings, quotienting may appear simple. The integers modulo n identify numbers that differ by a multiple of n. A group modulo a normal subgroup identifies elements lying in the same coset. A vector space modulo a subspace identifies vectors differing by an element of that subspace. A topological quotient identifies points according to a specified equivalence relation and equips the result with the quotient topology. Each construction forgets something, but not arbitrarily. The forgotten distinctions are exactly those authorized by the quotient policy.

The phrase “permitted forgetting” is important. Forgetting is not automatically lawful. If one quotients a group by a subset that is not a normal subgroup, the multiplication of cosets may fail to be well-defined. If one quotients an algebraic structure by a relation that is not a congruence, the operations may not descend. If one collapses topological points without regard to separation, compactness, or continuity conditions, the quotient may behave badly for the intended purpose. If one quotients a proof corpus by semantic similarity, theorem-critical differences may be erased. The issue is not whether collapse is possible, but whether the collapse preserves the relevant invariants.

Quotienting therefore belongs at the center of Admissibility Algebra. It is the mathematical operation by which policy-indexed identity becomes a new domain. Given a universe Ω and an admissibility policy π, the quotient Ω/π is the world obtained by treating π-equivalent things as the same. But this world is legitimate only if π is compatible with the structures one wants to retain. The quotient must be controlled by invariants, certificates, and substitution rules.

For mathematical AI, quotienting appears everywhere. An embedding model quotients language into vector neighborhoods. A retrieval system quotients theorem statements by similarity. A proof assistant quotients expressions by definitional equality or declared equivalence. A summarizer quotients a proof into a shorter explanation. A classifier quotients examples into types. A formalizer quotients informal variation into canonical formal objects. Each operation forgets distinctions. The question is whether the forgotten distinctions are admissible to forget.

This is why an AI system cannot be judged only by compression efficiency or retrieval speed. Compression may be destructive. A theorem statement compressed into an embedding may

lose hypotheses. A proof summarized into prose may lose dependencies. A category quotient may erase automorphism data. A homotopy quotient may preserve information that a naive orbit quotient destroys. A database merge may identify two mathematical objects that share a name but differ in definitions. These are quotient failures.

Admissibility Algebra demands that every quotient answer four questions. What distinctions are being collapsed? Which invariants survive? What certificate proves the collapse is lawful? What substitutions become licensed in the quotient world? If these questions cannot be answered, the quotient is not yet admissible. It may be a heuristic abstraction, but not a certified mathematical transformation.

8.2 The formula Ω/π

The formula Ω/π expresses a quotient domain formed by applying admissibility policy π to a base universe Ω. Here Ω may be a set, type, category, class of structures, proof corpus, theorem database, representation space, model state space, or mathematical context. The policy π specifies when two elements or objects of Ω are to be treated as identical for the purposes of the quotient. The resulting Ω/π is not merely smaller than Ω. It is Ω viewed through a permitted identity rule.

This notation generalizes familiar quotients. If Ω is the set of integers and π is congruence modulo n, then Ω/π is Z/nZ. If Ω is a group G and π identifies elements in the same coset of a normal subgroup N, then Ω/π is G/N. If Ω is a vector space V and π identifies vectors differing by an element of a subspace W, then Ω/π is V/W. If Ω is a topological space and π is an equivalence relation, then Ω/π is the quotient space with topology induced by the projection. If Ω is a class of algebraic structures and π is isomorphism, then Ω/π is a classification up to isomorphism.

The notation also applies to less elementary domains. If Ω is a category of models and π is categorical equivalence, Ω/π is a domain of models up to equivalence. If Ω is a space of gauge fields and π is gauge equivalence, Ω/π is a physical configuration space modulo representational redundancy. If Ω is a collection of formal proofs and π is proof irrelevance, normalization, or proof equivalence, Ω/π identifies proofs according to the selected proof policy. If Ω is a proof-search database and π is theorem-relevance equivalence, Ω/π clusters statements by admissibly preserved mathematical content.

The danger is that Ω/π may be written before π is adequate. A relation can always be imposed, but not every imposed relation preserves the structure one wants. In algebra, π must usually be a congruence. In topology, the quotient topology must support the intended continuity claims. In category theory, quotienting objects may require keeping morphisms, automorphisms, and coherence data. In homotopy theory, naive orbit spaces may lose stabilizer information

preserved by homotopy quotients. In mathematical AI, quotienting by embedding similarity may lose formal distinctions needed for proof.

Thus Ω/π is not only a notation. It is a demand for a policy certificate. To form Ω/π responsibly, the system must know the base domain Ω, the relation or transformation class π, the invariant family Iπ, the certificate type Cπ, the substitution scope Sπ, and the quotient construction rules. The quotient should also record what is lost. Controlled collapse requires an audit trail of discarded distinctions.

This formula is particularly useful for AI because it makes abstraction explicit. Every abstraction can be viewed as a quotient. A concept class is a quotient of examples. A summary is a quotient of a document. A theorem schema is a quotient of instances. A formal object may be a quotient of informal presentations. An embedding vector may be a quotient of text into semantic coordinates. An admissibility-aware AI should therefore ask, whenever it abstracts: what is Ω, what is π, and what has Ω/π forgotten?

The formula also supports multi-policy reasoning. The same Ω can have many quotients: Ω/π_1, Ω/π_2, Ω/π_3. A class of groups may be quotiented by cardinality, by isomorphism, by elementary equivalence, by representation type, or by Morita-like equivalence in a categorical setting. Each quotient serves different mathematics. An AI should not confuse them. It should know which quotient world it is operating in.

The expression Ω/π therefore captures the public meaning of Admissibility Algebra: identity is not merely found; it is formed by policy. A quotient is the world after policy has declared which differences no longer count.

8.3 Useful abstraction versus destructive collapse

Abstraction is useful when it forgets irrelevant distinctions while preserving the invariants required for the task. Collapse is destructive when it forgets distinctions that later reasoning needs. The difference is not psychological or aesthetic. It is policy-theoretic. A useful abstraction is a quotient Ω/π whose policy π preserves the relevant invariants. A destructive collapse is a quotient whose policy erases invariant content.

Mathematics advances by useful abstraction. Treating isomorphic groups as the same permits classification. Treating vector spaces up to dimension over a fixed field reveals a clean structural theorem. Treating spaces up to homotopy equivalence makes algebraic topology possible. Treating gauge-related fields as physically equivalent removes representational redundancy. Treating propositions up to logical equivalence can simplify proof. These are not losses of rigor. They are controlled forgettings.

But the same act, applied under the wrong policy, becomes destructive. Treating groups of equal cardinality as the same destroys group structure. Treating spaces with the same homology as the same may erase homotopy distinctions. Treating equivalent categories as strictly identical may break formal constructions sensitive to representatives. Treating proof sketches as proof objects destroys verification. Treating semantically similar theorem statements as interchangeable destroys hypotheses. Treating approximate numerical equality as exact equality destroys algebraic reasoning.

This is the central quotient problem for mathematical AI. AI systems are abstraction machines. They compress, cluster, summarize, embed, retrieve, classify, normalize, canonicalize, and generalize. These are quotient-like operations. Without explicit policies, the system may perform destructive collapse while appearing efficient. A short summary may be easier to read but false to the theorem. A vector cluster may retrieve nearby text but miss formal applicability. A canonicalization step may erase data required to reconstruct the original object. A theorem analogy may collapse domains that differ in a crucial invariant.

Useful abstraction requires three safeguards. First, the system must specify the intended invariants. Second, it must show that the quotient policy preserves them. Third, it must restrict downstream substitution to the quotient's valid scope. If any of these are missing, the abstraction should be marked heuristic rather than certified.

For example, suppose a mathematical AI builds a database of group examples. It may cluster groups by order. That quotient is useful for some tasks, such as enumeration by size. But it is destructive if used for group-isomorphism reasoning. The database must therefore label the quotient: Ω/cardinality, not Ω/group-isomorphism. A query asking whether two groups are the same must not use the cardinality quotient unless the requested policy is cardinality.

Likewise, a theorem-retrieval system may embed theorem statements and retrieve nearest neighbors. This is useful for search. But semantic proximity is not a certificate of applicability. The embedding quotient may suggest candidates, but proof-grounded reasoning requires a stronger policy: hypothesis matching, type matching, definitional compatibility, theorem instantiation, and formal proof construction. The system must distinguish retrieval quotient from proof admissibility.

Destructive collapse often hides in phrases such as "basically the same," "morally equivalent," "up to obvious identification," "similar argument," or "standard transfer." These phrases can be legitimate among experts, but for AI they must be unpacked. Which policy? Which invariant? Which certificate? Which scope? Without answers, the phrase may conceal an illicit quotient.

The conclusion is not that abstraction should be minimized. The conclusion is that abstraction should be certified. Mathematical AI needs abstraction to be useful, but it must learn the

difference between quotienting by a lawful policy and collapsing by superficial resemblance. The first is mathematics. The second is hallucination in quotient form.

8.4 Quotient objects, setoids, congruences, stacks, and homotopy quotients

Mathematics already contains many technologies for controlled collapse. Quotient objects, setoids, congruences, stacks, and homotopy quotients each address the same general problem: how can one identify things without losing the structure needed for valid reasoning?

A quotient object arises when an equivalence relation or congruence is imposed and a new object represents the collapsed domain. In sets, this is straightforward: elements become equivalence classes. In algebra, the relation must be compatible with operations. A group quotient requires a normal subgroup. A ring quotient requires an ideal. A module quotient requires a submodule. These compatibility conditions ensure that operations descend to equivalence classes. They are exactly admissibility conditions.

Setoids make equivalence explicit. Instead of relying on definitional equality alone, a setoid packages a carrier with an equivalence relation. Reasoning over setoids requires respecting that relation. This is a natural formal model for Admissibility Algebra because it refuses to treat equality as the only identity policy. A setoid says: here is the domain, and here is the relation under which elements count as the same. But Admissibility Algebra generalizes the idea by adding certificate types, invariants, substitution scope, obstructions, and policy comparison.

Congruences are equivalence relations compatible with structure. They are the algebraic condition that makes quotienting lawful. If $x \sim x'$ and $y \sim y'$, then operations must satisfy $f(x,y) \sim f(x',y')$, or the quotient operation is ill-defined. Congruence is therefore the local algebraic form of admissibility. It says the proposed identification respects the operations that matter. Without congruence, collapse destroys structure.

Stacks and related constructions arise when naive quotienting loses automorphism or descent data. In moduli problems, treating isomorphic objects as identical may erase information about their symmetries. A stack retains structured information about objects, isomorphisms, automorphisms, and gluing. From an admissibility viewpoint, stacks are refined quotient devices. They do not merely collapse by equivalence; they remember the ways equivalence occurs. This is an important lesson for AI: sometimes the certificate of equivalence is not disposable after quotienting. It remains part of the correct object.

Homotopy quotients provide a similar warning. If a group acts on a space, the naive orbit space may collapse points while losing stabilizer and higher homotopical information. The homotopy quotient retains more structure, often by replacing a strict quotient with a construction that remembers action data up to homotopy. This is precisely the kind of controlled collapse needed

when witnesses and symmetries matter. It shows that Ω/π may need to retain certificates, not merely equivalence classes.

Higher inductive types in homotopy type theory also belong here. They allow one to construct types by adding points, paths, and higher paths. Quotienting becomes a controlled introduction of identity witnesses. Instead of erasing distinctions silently, one explicitly adds paths identifying elements and then respects the resulting higher structure. This is deeply compatible with Admissibility Algebra because it treats identity formation as constructive and certificate-bearing.

For mathematical AI, these technologies offer design principles. A naive database quotient that merges all equivalent objects into one record may be too crude. It may need to store representatives, certificates, automorphisms, transport maps, coherence data, and quotient policy. A theorem database should know not only that two formulations are equivalent, but how they are equivalent, under what assumptions, and whether the equivalence is canonical. A proof corpus should not erase proof traces if later reasoning depends on construction.

This suggests that the admissibility database should support multiple quotient modes. Some quotients can be coarse and extensional. Others must be intensional, retaining witnesses. Some can identify objects completely for a task. Others should preserve a groupoid-like structure of objects and equivalences. Some should form ordinary equivalence classes. Others should form stacks, groupoids, or higher structures. The correct quotient type depends on π.

The general principle is that quotienting must preserve the evidence required by future substitution. If future reasoning needs only the class, the quotient may store the class. If future reasoning needs the witness, the quotient must store the witness. If future reasoning needs coherence among witnesses, the quotient must store higher structure. This is why quotient technology is central to mathematical AI.

8.5 Mathematical AI as a quotient-management system

A mathematical AI can be understood as a quotient-management system. It constantly decides which distinctions to preserve, which to collapse, which representatives to use, which equivalences to certify, and which quotient worlds to reason inside. This is true whether the system is generating proofs, retrieving lemmas, summarizing papers, formalizing statements, clustering examples, or explaining concepts.

In retrieval, the system quotients mathematical text by similarity. In formalization, it quotients informal variation into formal structure. In theorem proving, it quotients proof states by transformations that preserve provability. In simplification, it quotients expressions by equality or reduction. In classification, it quotients objects by isomorphism or another policy. In abstraction, it quotients examples by shared invariants. In explanation, it quotients formal detail into conceptual narrative. Each stage requires quotient discipline.

If the system lacks quotient discipline, it will hallucinate. It will retrieve a similar but inapplicable theorem. It will formalize a nearby but different statement. It will collapse examples into a false pattern. It will summarize away a necessary hypothesis. It will treat a quotient representative as canonical when it is not. It will erase automorphism data. It will merge definitions that differ in formal systems. These are not unrelated AI failures. They are quotient-management failures.

A quotient-aware mathematical AI should therefore maintain explicit records of quotient operations. When it collapses Ω to Ω/π, it should record the base domain, the policy, the preserved invariants, the collapsed distinctions, the certificate type, the substitution scope, and known obstructions. When it retrieves from a vector index, it should label the result as similarity-based, not proof-certified. When it canonicalizes an object, it should record what canonicalization policy was used. When it summarizes a proof, it should record which proof dependencies were preserved or omitted.

This architecture gives the system a way to separate heuristic quotienting from certified quotienting. Heuristic quotients are useful for search and creativity. Certified quotients are lawful for proof and substitution. The AI may use heuristic quotients to propose candidates, but it must pass through admissibility checks before asserting identity. Thus the system can remain creative without becoming unreliable.

The quotient-management view also explains why the database should include graph structure. Quotients relate representatives, equivalence classes, certificates, invariants, and policies. A relational table alone can store records, but a graph can express how objects collapse under different policies, how policies refine one another, how certificates transport, and how obstructions block stronger identifications. A vector index can support heuristic retrieval, but the graph and proof checker must govern admissibility. The three layers together implement quotient discipline.

A mature mathematical AI should be able to answer questions such as: under which quotient are these objects identical? What was forgotten in forming this quotient? Which invariants survive? Does the quotient preserve the operation? Is the relation a congruence? Are automorphisms retained? Is the quotient ordinary, homotopical, categorical, or stack-like? May this theorem be transported to the quotient? Does the proof require a representative? These questions are not peripheral. They are the core of trustworthy abstraction.

The conclusion of Section 8 is that quotienting is the mathematical form of abstraction, and abstraction is the central operation of intelligence. But abstraction without admissibility is destructive collapse. Mathematical AI must therefore become explicit about its quotients. It must know when it is operating in Ω, when in Ω/π, what π permits, what π forgets, and what certificates are required to move between representatives and quotient objects. In this sense,

proof-grounded AI is not merely a theorem generator. It is a disciplined manager of controlled collapse.

9. Proof Assistants as Admissibility Engines

9.1 Lean, Coq, Agda, Isabelle, and proof-object discipline

Proof assistants are the natural first machines for Admissibility Algebra because they already enforce a disciplined boundary between assertion and certificate. Lean, Coq, Agda, Isabelle, and related systems do not accept a theorem because it sounds plausible, resembles a known result, or follows a familiar rhetorical pattern. They require a formal object, derivation, or proof script that can be checked according to the rules of the system. This makes them primitive admissibility engines.

The deepest value of proof assistants is not merely that they automate mathematics. It is that they refuse unauthorized transformation. A rewrite must be justified. A theorem application must match its hypotheses. A term must have the declared type. A definition must elaborate. A proof must close all goals. A quotient must satisfy the required formation and elimination rules. A coercion must be declared. An equality must have the correct type. These requirements enforce local admissibility at every step.

This proof-object discipline is precisely what current generative AI often lacks. A language model may generate a proof-shaped paragraph, but the paragraph is not itself a certificate unless it can be translated into a proof object or otherwise checked. A proof assistant forces the distinction. The mathematical statement is a type, proposition, sequent, or formula. The proof is an inhabiting object, derivation, tactic trace, or kernel-checkable artifact. The assertion and the certificate are separated.

Lean and Coq are especially instructive because their foundations make proof terms central. A theorem is certified by a term inhabiting the corresponding proposition. Tactic scripts are not the final authority; they are programs that construct proof terms. The kernel checks the resulting object. This is admissibility in operational form: no matter how clever the tactic, no theorem enters the library without a valid certificate.

Agda emphasizes dependent types and constructive structure, making transport, equality, and representation choices explicit. This is valuable for Admissibility Algebra because many substitution errors become visible when types depend on the terms being replaced. What informal mathematics treats as obvious replacement may require explicit transport. Agda-like discipline teaches that identity witnesses are not decorative; they can be computationally and structurally active.

Isabelle, with its strong tradition of structured proof and multiple object logics, shows another aspect of admissibility: the relation between formal context, proof method, and trusted kernel. Its Isar language emphasizes human-readable structured proof, while the underlying system still enforces formal validity. This is important for mathematical AI because future systems must serve both machines and humans: proof objects for verification, explanations for understanding.

Proof assistants therefore instantiate the primitive judgment $\Gamma \vdash X \equiv_\pi Y : C$ in many local forms. Γ appears as the formal context: variables, hypotheses, imports, typeclass instances, definitions, locale assumptions, universe parameters, and available lemmas. X and Y may be terms, propositions, structures, definitions, or proof states. π may be definitional equality, propositional equality, rewriting by theorem, equivalence, isomorphism, setoid equality, quotient equality, or transport along identity. C is the proof term, tactic trace, derived theorem, equivalence object, or kernel-checked transformation.

The point is not that proof assistants already contain all of Admissibility Algebra. They do not. They enforce many admissibility constraints, but they do not necessarily index all sameness policies explicitly. They may check a rewrite without explaining the broader policy hierarchy. They may formalize isomorphism, equivalence, quotienting, and transport, but not store them as a unified admissibility graph. They may prove a theorem without recording how that theorem should guide future AI substitution. Admissibility Algebra generalizes and organizes what proof assistants already practice locally.

For mathematical AI, proof assistants should therefore be treated as certificate authorities. A neural model may propose. A retrieval system may suggest. A database may cluster. But a proof assistant can check whether a formal transformation is admissible within its jurisdiction. The long-term architecture is not pure neural generation and not pure proof search. It is neural proposal under proof-assistant discipline.

9.2 Formal libraries as admissibility corpora

Formal libraries are not merely collections of theorems. They are corpora of certified transformations. Every definition, theorem, instance, rewrite lemma, equivalence, isomorphism, quotient construction, coercion, and proof dependency records a fragment of admissibility. A library such as mathlib is therefore not only a mathematical encyclopedia; it is a machine-readable history of permitted identity transformations.

This is crucial for training mathematical AI. Ordinary mathematical text contains enormous conceptual richness, but much of its admissibility structure is implicit. Formal libraries expose more of that structure. They record the exact hypotheses under which a theorem applies. They distinguish definitional equality from proved equality. They track namespaces and typeclass

assumptions. They encode algebraic structures, morphisms, equivalences, quotient types, coercions, and rewrite rules. They preserve the dependency chain from statement to certificate.

For Admissibility Algebra, the first task is to mine formal libraries for existing sameness policies. These include equality, equivalence relations, setoids, isomorphisms, equivalences of types, equivalences of categories, homeomorphisms, diffeomorphisms, ring equivalences, group equivalences, linear equivalences, order isomorphisms, quotient relations, congruences, rewriting lemmas, normalization procedures, and transport mechanisms. Each is a candidate policy π.

The second task is to extract certificate forms. A formal library does not merely say that two structures are related. It defines the data required to witness the relation. A group isomorphism has fields or properties. A linear equivalence has maps and inverse laws. A homeomorphism has continuity data. A category equivalence has functorial and natural isomorphism data. A quotient relation has recursors and soundness conditions. These formal definitions reveal what each policy considers an admissibility certificate.

The third task is to extract invariants and obstruction patterns. Libraries contain theorems saying that certain properties are preserved under isomorphism, equivalence, homeomorphism, equivalence of categories, or other policies. They also contain theorems that distinguish objects by invariants. If one object has a property invariant under π and the other does not, admissibility under π fails. These preservation and non-preservation theorems are the raw material for an obstruction database.

The fourth task is to extract substitution scopes. Some theorems are invariant under isomorphism. Some require equality. Some transport over equivalence. Some require stronger structure. Some are declared as simp lemmas and participate in automated rewriting. Some are available only under particular typeclass assumptions. This tells the AI which contexts permit replacement. The library is already full of scoped substitution data, though it is not always labeled in those terms.

Formal libraries also provide negative lessons. A theorem may be difficult to formalize because the informal statement suppresses choices, coherence, or side conditions. Such failures are valuable. They reveal where informal admissibility relies on expert convention. A mathematical AI trained only on successful proofs may miss these boundaries. A mature admissibility corpus should include failed formalization attempts, theorem application failures, missing hypotheses, and rejected rewrites as obstruction data.

The formal library should therefore be reorganized into an admissibility corpus. The standard theorem database asks: what theorem has this name, statement, and proof? The admissibility corpus asks: what transformation does this theorem license, under which policy, preserving

which invariants, with which certificate, and in which substitution scope? This recoding turns formal libraries into training data for proof-grounded AI.

Such a corpus is also valuable for human mathematics. It would show when two concepts are definitionally equal, propositionally equivalent, isomorphic, equivalent, or merely related. It would reveal policy ladders across domains. It would make quotient assumptions explicit. It would show how a result moves from one context to another. It would expose where a theorem is robust under abstraction and where it depends on representation.

The immediate conclusion is practical. Before building a large mathematical AI, one should build a mathematical admissibility corpus from formal libraries. The corpus should not merely index theorem statements. It should index policies, certificates, invariants, obstructions, quotient rules, and substitution scopes. That corpus becomes the memory of admissible mathematical transformation.

9.3 The theorem prover as certificate checker

The theorem prover should be viewed as a certificate checker, not as the whole intelligence. This distinction matters. A theorem prover does not need to originate every idea. It need not generate every conjecture. It need not choose every analogy. Its essential role in the admissibility architecture is to check whether a proposed formal transformation is valid under its rules. It is the court of final appeal for formal certificates within its jurisdiction.

In this architecture, the neural model proposes candidate transformations. Retrieval supplies relevant theorems. A graph database supplies policy relations and dependency structure. A vector index supplies semantic neighbors. The theorem prover checks the certificate. If the certificate is accepted, the admissibility judgment can be recorded as verified. If it fails, the failure can be classified as an obstruction, missing hypothesis, type mismatch, elaboration failure, or incomplete search.

This division of labor protects both creativity and rigor. Neural systems are useful because they search broadly and suggest possibilities. Proof assistants are useful because they refuse invalidity. If one relies only on neural generation, hallucination enters. If one relies only on brute formal search, exploration may be too narrow. The admissibility engine combines them: generative below, verificational above.

The theorem prover also enforces context Γ. A proof script that succeeds in one context may fail in another because imports, instances, assumptions, or definitions differ. This is not a nuisance; it is admissibility information. The context determines the jurisdiction. If a theorem application fails because a required algebraic structure is absent, the system has learned that the proposed substitution was not authorized in Γ. If a rewrite fails because the equality is not definitional or not available, the system has learned a policy boundary.

Certificate checking should therefore produce structured outputs. A successful check should return the theorem proved, the proof object or proof trace, the dependencies used, the context, and the policy under which the transformation was valid. A failed check should return the failed goal state, the attempted rule, missing hypotheses, type errors, unification failures, unresolved instances, or other diagnostic information. These outputs are not merely logs. They are training data.

The theorem prover is especially important for distinguishing proof from proof-shaped explanation. A language model may say that a result follows by induction. The theorem prover asks which induction principle, on which variable, with which base cases and induction hypotheses, and whether the goal actually closes. A model may say that a diagram commutes naturally. The theorem prover asks for the components and naturality squares. A model may say two expressions are equal by simplification. The theorem prover asks whether simplification under the current rewrite rules proves the equality.

This discipline turns mathematical AI from a text generator into a certificate-producing system. The final result is not merely an answer but a checked artifact. For formal mathematics, the strongest certificate is kernel acceptance. The proof assistant kernel is small relative to the language around it and is designed to be trusted. Thus, when possible, admissibility records should point to kernel-checked proof terms rather than only to generated prose or tactic scripts.

However, theorem provers also have limits. Formalization is labor-intensive. Libraries are incomplete. Some mathematical claims are difficult to encode. Some certificates may be informal, computational, or external to the current formal system. A theorem prover can check within its world, not all worlds. Therefore the admissibility architecture should record the formal system as part of Γ and avoid treating one prover's jurisdiction as universal mathematical truth.

The theorem prover is a certificate checker. It is not the source of all policy, but it enforces many policies with unmatched precision. Mathematical AI should treat it as the central verifier in the admissibility loop.

9.4 Tactic scripts as transformation traces

Tactic scripts are not merely commands that make a proof assistant accept a theorem. They are traces of attempted transformation. Each tactic changes the proof state by applying a rule, simplifying a goal, introducing variables, splitting cases, rewriting terms, invoking lemmas, constructing witnesses, or discharging obligations. From the viewpoint of Admissibility Algebra, a tactic script is a sequence of admissibility attempts.

This makes tactic scripts highly valuable training data. A completed tactic proof records a successful path from initial goal to certified theorem. Each step is a local transformation

licensed by the proof assistant. A failed tactic attempt records a blocked transformation: a missing hypothesis, wrong goal shape, unavailable instance, nonmatching lemma, or invalid rewrite. Both successes and failures teach the AI the boundaries of admissible proof-state movement.

A tactic trace can be represented as a sequence:

$\Gamma_0 \vdash \text{Goal}_0$
Tactic_1 transforms Goal_0 into Goal_1
Tactic_2 transforms Goal_1 into Goal_2
...
Tactic_n closes all goals

Each transition has a policy. It may be introduction, elimination, rewriting, simplification, normalization, induction, case splitting, exact proof term application, theorem instantiation, or search. The tactic is admissible only if the proof assistant accepts the state transition. Thus a tactic script is a concrete operational witness of proof transformation.

For mathematical AI, tactic traces are better than final theorems alone. A final theorem says what is true. A tactic trace shows how proof-state identity was transformed into closure. It reveals common proof moves, dependency structures, side conditions, and local admissibility patterns. It shows where rewriting works, where induction applies, where simplification is sufficient, and where human guidance is needed. It also reveals bottlenecks: places where models often choose the wrong lemma, wrong induction variable, wrong rewrite direction, or wrong policy.

Tactic scripts also expose the distinction between heuristic and certified transformation. A tactic may be suggested by a neural model, but until the proof assistant runs it, its admissibility is unknown. Once accepted, the tactic's result is certified by the proof state transition and ultimately by the constructed proof object. If rejected, the failure becomes structured obstruction data. This provides a natural reinforcement signal for mathematical AI: successful tactic transitions are positive examples; failed transitions are negative or diagnostic examples.

However, tactic scripts are not always ideal certificates by themselves. They may be brittle, dependent on library state, sensitive to simplifier configuration, or opaque in their effect. The true certificate is the proof object produced or the theorem accepted by the kernel. The tactic script is best understood as a transformation trace leading to that certificate. It is valuable for learning and explanation, but the kernel-checked result remains the strongest formal witness.

Tactic traces should therefore be stored with metadata. The record should include the initial goal, final goal state, intermediate states if available, tactics used, lemmas invoked, imports, library version, formal system version, elapsed time, success or failure, and generated proof

object if accessible. For admissibility purposes, it should also classify the policy of each step: rewrite, transport, simplification, induction, case split, equivalence application, quotient recursor, coercion, or theorem application.

This classification turns proof scripts into admissibility sequences. A future AI can learn not merely that a tactic worked, but what kind of transformation it performed. It can learn that a theorem was used as a rewrite, that an isomorphism was used for transport, that a quotient recursor required well-definedness, or that a simplification closed a definitional equality. Such learning is closer to mathematical understanding than pattern imitation.

Tactic scripts also support human-facing explanations. A system can translate a tactic trace into prose while preserving the proof's dependency structure. But the explanation should remain linked to the trace. Otherwise the AI may produce a pleasant narrative that loses proof identity. The trace anchors explanation to certificate.

The conclusion is that tactic scripts should be treated as transformation traces inside the admissibility corpus. They are not merely code. They are records of how mathematical identity was lawfully transformed until a proof closed.

9.5 What proof assistants already know, and what they do not yet index

Proof assistants already know a great deal about admissibility. They know whether terms type-check. They know when definitions reduce. They know whether a proof object inhabits a proposition. They know which hypotheses are available. They know when a rewrite applies. They know when a theorem can be instantiated. They know whether quotient recursors satisfy well-definedness conditions. They know when coercions and typeclass instances can be resolved. They know when a proof state has been closed.

They also encode many mathematical policies. Equality, equivalence relations, setoids, isomorphisms, equivalences, homeomorphisms, linear equivalences, order isomorphisms, quotient types, congruences, transport functions, extensionality principles, simplification rules, and rewriting systems are all present in modern formal libraries. In this sense, proof assistants are already rich admissibility environments.

But proof assistants do not necessarily index this knowledge in the form needed by mathematical AI. They may know that a proof is valid without classifying which admissibility policy each step uses. They may contain many isomorphism notions without organizing them into a policy lattice. They may support quotient constructions without recording, in a searchable way, what distinctions were collapsed and which invariants survive. They may accept a rewrite without labeling its substitution scope for future retrieval. They may contain preservation theorems without linking them to a general invariant database.

The missing layer is not validity but organization. Formal libraries are built for formal proof development, not necessarily for AI-readable admissibility analysis. A human or model can search the library, but the library does not automatically present itself as a graph of policies, certificates, invariants, obstructions, quotients, and scoped substitutions. Admissibility Algebra proposes this re-indexing.

For example, a proof assistant may define many structures called equivalences in different namespaces. A mathematical AI needs to know how they relate. Which are stronger? Which forget to which? Which support rewriting? Which preserve which invariants? Which are canonical? Which require choices? Which support transport into which contexts? Which have formal certificates already available? This is policy graph information.

Similarly, a proof assistant may contain many theorems of the form "property P is preserved under equivalence E." These are invariant-preservation records. An admissibility index should extract and organize them. It should allow a query: under policy π, is property P transportable? If yes, what theorem certifies it? If no, is there an obstruction? If unknown, what related policies exist?

Proof assistants also under-index failures. Failed proof attempts, abandoned formalizations, rejected rewrites, and missing hypotheses are usually not preserved as first-class data. But for AI, these are valuable. They teach what not to do. They expose admissibility boundaries. A corpus containing only successful proofs gives an incomplete picture of mathematical practice. A richer admissibility corpus should preserve structured failure traces.

Another gap is informal-formal correspondence. A proof assistant may certify a formal theorem, but it does not by itself certify that this theorem faithfully represents an informal source. That requires a formalization policy. The objects are an informal statement and a formal encoding; the certificate must show semantic fidelity. Proof assistants check the formal side, but Admissibility Algebra must also represent the translation policy connecting informal and formal mathematics.

Finally, proof assistants do not eliminate the need for multiple policies. The kernel may enforce its own equality and proof rules, but mathematicians reason across many equivalence levels. An AI assistant must know when the formal system requires equality even though the human statement is invariant only up to isomorphism, or when a category-theoretic equivalence must be transported through formal structures that demand explicit data. This is the practical boundary between human mathematical convention and machine formalization.

The conclusion of Section 9 is that proof assistants are indispensable but incomplete admissibility engines. They provide certificate checking, context discipline, proof-object rigor, and formal libraries. What they lack is an explicit AI-facing admissibility index: policies as first-

class objects, invariants as searchable requirements, certificates as typed witnesses, obstructions as negative records, quotients as controlled collapses, and substitution scopes as operational permissions. Building that index is the next step toward proof-grounded mathematical AI.

10. The Admissibility Database for Mathematical AI

10.1 Objects, policies, invariants, certificates, obstructions, quotients

A mathematical AI built on Admissibility Algebra requires a database whose primitive units are not merely documents, theorem statements, embeddings, or generated outputs. It requires a database of admissibility. The core entities are mathematical objects, policies, invariants, certificates, obstructions, quotients, and judgments. These are the structural components by which a system learns what may count as the same.

A MathematicalObject is anything that can enter a formal or informal mathematical comparison: a group, ring, module, category, type, space, theorem, proof fragment, definition, diagram, formal declaration, proof term, tactic state, algorithm, normal form, or representation. Objects should be stored with their context, source, formal encoding when available, informal description, namespace, dependencies, and version information. The same mathematical object may have many presentations, and those presentations should not be prematurely collapsed.

A Policy is a rule of permitted sameness. Examples include literal equality, definitional equality, propositional equality, isomorphism, equivalence of categories, homotopy equivalence, quotient equivalence, setoid equivalence, Morita equivalence, gauge equivalence, logical equivalence, semantic equivalence, and approximation within certified error. Each policy should specify its domain, certificate type, invariant family, substitution scope, strength relations to other policies, and failure modes. A policy is not a comment attached to equality; it is a first-class object.

An Invariant is a property, structure, relation, or behavior required to survive under a policy. Invariants may include cardinality, dimension, rank, order, kernel, spectrum, homology, universal property, proof obligation, type correctness, definitional reduction, commutativity, continuity, functoriality, naturality, or semantic content. An invariant may be complete for a classification problem or merely obstructive. The database should distinguish complete invariants, partial invariants, computable invariants, proof-based invariants, and heuristic invariants.

A Certificate is the witness that a claimed admissibility judgment holds. It may be a proof object, explicit isomorphism, equivalence data, homotopy data, quotient relation witness, computation

trace, formal theorem, tactic proof, normalization result, or verified derivation. The certificate should include its type, formal status, checker, dependencies, source, construction method, and whether it is canonical or choice-dependent. Without certificate data, the system stores claims but not admissibility.

An Obstruction is a certified reason why admissibility fails under a given policy. It may be an invariant mismatch, missing hypothesis, failed congruence, counterexample, type error, failed transport, non-preserved structure, or formal rejection. Obstructions are not failed guesses. They are negative witnesses. A mature database should store obstructions with the same seriousness as certificates because they define the boundary of lawful substitution.

A Quotient is the controlled collapse Ω/π formed by applying policy π to a domain Ω. The quotient record should identify what was collapsed, what was preserved, what certificate type governs membership, whether witness data are retained, and which substitutions are valid in the quotient world. Quotients are the database form of abstraction. Without explicit quotient records, the AI cannot know whether it is reasoning in Ω, Ω/π, or an unlicensed compression of Ω.

Together these entities form the grammar of mathematical admissibility. They allow the AI to answer not merely “what theorem is relevant?” but “under what policy may this object replace that object, preserving which invariant, with what certificate, and what would obstruct the replacement?” This is the database needed for proof-grounded transformation.

10.2 The admissibility judgment as the atomic database record

The atomic database record is the admissibility judgment:

$\Gamma \vdash X \equiv_\pi Y : C$

or, on failure:

$\Gamma \nvdash X \equiv_\pi Y$ because $O_\pi(X,Y)$

This record is more fundamental than a theorem statement alone because it captures the operation mathematical AI actually needs to perform: deciding whether one object, representation, claim, proof, or construction may stand in for another under a named policy. A theorem database stores facts. An admissibility database stores permissions.

The judgment record should include the context Γ, the compared objects X and Y, the policy π, the result state, the certificate C if positive, the obstruction O if negative, verification status, source provenance, formal system, library version, dependency graph, and substitution scope. It should also record whether the judgment is exact, approximate, heuristic, formal, informal, machine-checked, human-annotated, or undecided.

The result state should not be binary. At minimum it should include admissible, inadmissible, undecided, and uncertified. Admissible means a certificate has been verified. Inadmissible means an obstruction has been verified. Undecided means neither certificate nor obstruction is known. Uncertified means a claim has been proposed but not yet checked. These distinctions are essential because mathematical AI must not confuse failure to search with disproof, or plausibility with proof.

A single pair X,Y should usually have many judgment records under different policies. For example, $\mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ are admissibly the same under cardinality but inadmissible under group isomorphism. Two categories may be admissible under equivalence but inadmissible under strict isomorphism. Two spaces may be admissible under homology-profile comparison but undecided or false under homotopy equivalence. The database must therefore support multi-policy identity, not one global equivalence label.

The judgment record should also include directionality when relevant. Some transformations are symmetric, such as many equivalence relations. Others are directed, such as embeddings, quotient maps, approximations, reductions, simulations, interpretations, forgetful functors, and proof refinements. A directed admissibility record may say that X can be safely transformed into Y, but not that Y can reconstruct X. This matters for summaries, quotienting, formalization, computation, and abstraction.

Substitution scope should be attached directly to the judgment. The record must state what the certificate licenses. A group isomorphism licenses group-theoretic transport. A homotopy equivalence licenses transport of homotopy invariants. A quotient certificate licenses reasoning in the quotient only through well-defined operations. A formal proof certificate licenses the theorem in the formal system and context in which it was checked. Without scope, the system may overuse a valid certificate.

The admissibility judgment also supports auditability. If the AI later uses Y in place of X, the system should be able to point back to the record that licensed the substitution. If the substitution fails, the system should reveal whether the failure came from wrong policy, missing certificate, invalid transport, outdated dependency, or context mismatch. This turns mathematical AI into a traceable transformation system rather than a black-box generator.

In implementation terms, the admissibility judgment is the database row, graph edge, vector-associated record, and proof trace anchor around which all other data organize. It is the atomic unit of admissible mathematical memory.

10.3 Relational, graph, and vector layers

No single database structure is sufficient for mathematical admissibility. A useful architecture requires at least three layers: a relational layer, a graph layer, and a vector layer. Each serves a different purpose, and each corresponds to a different aspect of mathematical AI.

The relational layer is the system of record. It stores stable entities with explicit fields: objects, policies, invariants, certificates, obstructions, quotient records, judgments, source documents, formal systems, proof traces, model outputs, and verification status. It supports exact queries, versioning, provenance, constraints, and audit trails. PostgreSQL or an equivalent relational system is appropriate for this layer because admissibility data must be durable, structured, and correctable.

The graph layer stores relationships. Mathematics is dependency-rich. Theorems depend on definitions, lemmas, proof terms, imports, typeclass instances, constructions, and previous certificates. Policies refine other policies. Invariants are preserved by some policies and not by others. Certificates compose. Obstructions block stronger judgments. Quotients collapse representatives. Formalizations translate informal objects into formal encodings. These relations are naturally graph-like. A graph layer allows the AI to navigate from a proposed substitution to relevant policies, invariants, certificates, and failure cases.

The vector layer supports heuristic retrieval. Mathematical language is varied. The same concept appears under different notations, names, conventions, and explanatory styles. A vector index helps retrieve semantically related theorem statements, definitions, papers, proof sketches, examples, and prior judgments. But the vector layer must be treated as heuristic, not authoritative. Similarity retrieval proposes candidates; it does not certify admissibility. The vector layer is a search engine, not a judge.

These three layers correspond to three modes of intelligence. The relational layer remembers verified facts. The graph layer understands structural connection. The vector layer finds plausible neighbors. Mathematical AI needs all three, but they must not be confused. A vector match is not a certificate. A graph path is not automatically a proof. A relational record is only as good as its verification status. The admissibility engine coordinates the layers by routing proposals through policy and certificate checks.

For example, suppose the AI is asked whether two mathematical objects are the same. The vector layer may retrieve similar objects and relevant discussions. The graph layer may identify policies under which such objects are usually compared. The relational layer may store existing judgments. If no judgment exists, the system may search for certificates or obstructions. A proof assistant may verify the result. The new judgment is then stored in the relational layer, linked in the graph, and embedded in the vector layer for future retrieval.

This layered architecture also prevents destructive collapse. A purely vector system may merge objects that are semantically close but mathematically distinct. A purely relational system may be too rigid and fail to find relevant analogies. A purely graph system may capture dependencies but not linguistic variation. The three-layer system allows heuristic exploration while preserving formal admissibility boundaries.

The graph layer should include typed edges such as preserves, violates, certifies, obstructs, refines, weakens, quotients, transports, depends_on, formalizes, rewrites, instantiates, generalizes, specializes, and equivalent_under. These edge labels matter because the same two objects may be connected in many ways. An AI must distinguish “similar to” from “isomorphic to,” “used by” from “proves,” “quotients to” from “embeds in,” and “preserves invariant” from “violates invariant.”

The relational, graph, and vector layers therefore implement the database analogue of Admissibility Algebra. The relational layer stores judgments. The graph layer stores admissibility structure. The vector layer supports discovery. Proof assistants and external checkers provide certificate validation. Together they create a machine memory for policy-indexed mathematical identity.

10.4 Retrieval of relevant policies and invariants

When a mathematical AI confronts a proposed transformation X → Y, its first serious task is not to produce prose. It must retrieve the relevant policies and invariants. The question “Are X and Y the same?” must become “Under which policies might X and Y be compared, and what invariants would those policies require?”

Policy retrieval depends on object type, context, task, and intended use. If X and Y are finite sets, cardinality and bijection policies may be relevant. If they are groups, group isomorphism, homomorphism, quotient, subgroup, and representation policies may be relevant. If they are topological spaces, homeomorphism, homotopy equivalence, weak equivalence, and homology comparison may be relevant. If they are categories, strict isomorphism, equivalence of categories, adjunction, functorial comparison, and Morita-like policies may be relevant. If they are theorem statements, logical equivalence, implication, formalization fidelity, definitional equality, and proof equivalence may be relevant.

Invariant retrieval then asks what must be preserved under each candidate policy. For group isomorphism, retrieve order, operation, identity, inverses, element-order spectrum, subgroup structure, and other group invariants. For homeomorphism, retrieve topological invariants. For homotopy equivalence, retrieve homotopy-invariant properties. For formal theorem equivalence, retrieve hypotheses, quantifiers, domains, assumptions, definitions, and

conclusion. For proof transformation, retrieve proof obligations, dependencies, and tactic-state requirements.

This retrieval process must distinguish decisive invariants from suggestive invariants. Some invariants classify in a particular domain. Others only obstruct. Same dimension classifies finite-dimensional vector spaces over a fixed field up to isomorphism, but same order does not classify finite groups. Same homology groups do not generally classify spaces up to homotopy equivalence. Same characteristic polynomial does not always classify matrices up to similarity. A mathematical AI must learn which invariants are complete, which are partial, and which are merely heuristic.

The database should therefore store invariant status. An invariant record should indicate whether it is preserved under π, whether it is complete for some subclass, whether mismatch obstructs admissibility, whether match suggests but does not certify admissibility, and whether computation is available. This prevents the AI from overusing partial invariants. It can say: "these invariants agree, but they do not certify equivalence," or "this invariant differs, so equivalence under π is impossible."

Retrieval should also locate known certificates and obstructions. If a previous judgment exists for X,Y,π, the system can reuse it. If an isomorphism has already been constructed, it can retrieve the certificate. If an obstruction has already been established, it can avoid repeated search. If a stronger policy certificate exists, it may induce a weaker policy certificate by a forgetful map. If a weak policy obstruction exists, it may block stronger policies. Policy retrieval and invariant retrieval therefore interact with the policy lattice.

For theorem proving, relevant-policy retrieval can guide proof search. If the goal requires equality but only isomorphism is available, the system knows that transport or extensionality may be required. If a quotient is involved, it knows to search for well-definedness lemmas. If a rewrite fails, it can ask whether a setoid rewrite, congruence lemma, or equivalence transport is needed. This makes proof search policy-aware instead of tactic-pattern-only.

For informal mathematical writing, retrieval of policies and invariants allows better explanations. Rather than saying "these structures are essentially the same," the AI can say "they are isomorphic as groups, but not identical as presentations; the isomorphism preserves multiplication and element orders, so group-theoretic properties transport." This is a more precise and educational output.

The retrieval problem is therefore central. Mathematical AI should not retrieve only relevant text. It should retrieve the governing identity policy and the invariants that policy protects. Without that retrieval, all subsequent generation is underdetermined.

10.5 From formal corpus to machine-learnable admissibility

The purpose of the admissibility database is not only storage. It is to create a machine-learnable corpus of mathematical permission. A model trained on ordinary text learns how mathematicians talk. A model trained on proof scripts learns some patterns of formal derivation. A model trained on admissibility records learns which transformations are allowed, under which policies, preserving which invariants, with what certificates, and where substitutions fail.

A training example should therefore contain more than input and output. It should include context Γ, objects X and Y, policy π, invariant family $I\pi$, certificate C if admissible, obstruction O if inadmissible, result state, substitution scope, and source provenance. The model should see positive examples, negative examples, undecided examples, and uncertified proposals. It should learn the difference between plausible, proved, disproved, open, and policy-dependent.

Contrastive training is especially important. The same pair of objects should appear under multiple policies. $\mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ should be same under cardinality and different under group isomorphism. Equivalent categories should be same under categorical equivalence and not necessarily same under strict isomorphism. Homotopy equivalent spaces should be same under homotopy policy but not necessarily under homeomorphism. The model must learn that identity judgments change with π.

The corpus should also include near-misses. A theorem that almost applies but lacks a hypothesis. A quotient that almost works but fails congruence. A formalization that almost matches the informal theorem but changes a quantifier. A proof sketch that almost proves the statement but uses a stronger assumption. Near-misses teach the model that mathematical boundaries are often thin and decisive. They are the antidote to fluent hallucination.

Formal libraries supply the strongest seed data because they contain checked certificates. But they should be augmented with informal-formal alignments, failed proof attempts, theorem retrieval logs, tactic failures, human annotations, and paper-derived examples. The goal is not only to learn formal proof search. The goal is to learn admissibility across the pipeline from informal concept to formal certificate.

The model trained on this corpus should perform several tasks. Given X,Y, it should suggest relevant policies. Given π, it should identify required invariants. Given an invariant mismatch, it should produce an obstruction. Given a proposed certificate, it should check whether the certificate type matches the policy. Given a failed proof attempt, it should classify the failure. Given a broad claim of sameness, it should refine the claim into policy-indexed judgments. Given a theorem-like output, it should mark what remains uncertified.

This transforms mathematical AI from a next-token engine into an admissibility assistant. The neural model still proposes, retrieves, generalizes, and explains. But its training target includes the policy structure that governs mathematical identity. It learns not only what mathematicians say, but what mathematicians are allowed to substitute.

The admissibility corpus also creates a path to incremental development. The first version need not cover all mathematics. It can begin with finite groups, vector spaces, elementary topology, basic category theory, and formal theorem examples. It can store a small number of policies and invariants. It can demonstrate multi-policy judgments. It can connect to Lean or another proof assistant for certificate checking. Once the architecture works, the corpus can expand.

The long-term result is a new kind of mathematical memory. Not merely a library of facts. Not merely a search index. Not merely a proof repository. A memory of admissible transformations: what may count as the same, where, why, and with what witness. Such a memory is the proper training ground for proof-grounded mathematical AI.

The conclusion of Section 10 is direct. Mathematical AI needs an admissibility database because its central operation is transformation under identity constraints. The database must store objects, policies, invariants, certificates, obstructions, quotients, and judgments. It must combine relational precision, graph structure, vector retrieval, and proof-assistant verification. Once built, it becomes the corpus from which AI can learn not just mathematical language, but mathematical permission.

12. Toward Future AI: From Mathematical Admissibility to General Trust

12.1 Mathematics as the clean first domain

Mathematics is the clean first domain for Admissibility Algebra because its policies of sameness are already partially formalized. Equality, isomorphism, equivalence, homotopy equivalence, definitional equality, propositional equality, quotient equality, congruence, setoid equivalence, and formal proof all have established meanings. Their certificates can often be written down. Their invariants can often be named. Their obstructions can often be proved. Their substitution scopes can often be stated precisely. This makes mathematics the correct training ground before exporting admissibility to broader AI.

The difficulty of general AI trust lies partly in the vagueness of ordinary identity. What counts as the same meaning, the same intention, the same memory, the same person, the same plan, the same evidence, the same story, or the same moral constraint is often disputed, contextual, and socially negotiated. Mathematics is not free from interpretation, but it has a stronger discipline

of explicit definition. This allows an admissibility system to be built where policies can be formal, certificates can be checked, and failures can be diagnosed.

Mathematics also contains many levels of sameness in one domain. Literal equality, isomorphism, categorical equivalence, homotopy equivalence, Morita equivalence, logical equivalence, definitional equality, and quotient equivalence all appear as legitimate but distinct identity policies. Thus mathematics offers a rich training environment for policy-indexed reasoning. The AI can learn that sameness is not singular. It can learn that the correct answer depends on π. It can learn that the same objects may be admissibly identical under one policy and inadmissibly identified under another.

Proof assistants make mathematics uniquely valuable. They provide external certificate checking. A language model may propose a proof, but Lean, Coq, Agda, Isabelle, or another formal system can reject or accept the certificate within a defined jurisdiction. This gives training data with unusually sharp labels: accepted proof, failed proof, missing hypothesis, type error, invalid rewrite, unresolved goal, or kernel-checked theorem. Such feedback is much stronger than ordinary human preference data.

Mathematics is also an ideal domain because it makes obstruction meaningful. In many domains, a failed claim may be disputed or probabilistic. In mathematics, a failed invariant can decisively block a policy. If element-order structure differs, two groups are not isomorphic. If homology differs, two spaces are not homotopy equivalent. If a relation is not a congruence, a quotient operation is not well-defined. These obstructions teach the AI how to say no with reason.

The choice of mathematics as the first domain does not narrow the ambition. It protects it. A system that cannot distinguish equality from isomorphism, isomorphism from equivalence, equivalence from analogy, and proof from explanation is not ready to manage scientific evidence, legal identity, memory continuity, or real-world action. Mathematics supplies the training ground where admissibility can become exact before it becomes general.

The first objective is therefore not a universal trustworthy AI. It is a mathematical admissibility assistant. Such a system would identify policies, retrieve invariants, construct certificates, detect obstructions, manage quotients, and guide proof-grounded substitution. Once this discipline exists in mathematics, the architecture can be exported cautiously to broader domains.

12.2 Hallucination as policy failure

Hallucination in AI is usually described as false generation: the model produces an incorrect claim, invented citation, nonexistent theorem, invalid proof, or unsupported explanation. Admissibility Algebra gives a deeper diagnosis. Hallucination is policy failure. The system

performs a transformation $X \to Y$ while behaving as if $X \equiv_\pi Y$, but the policy π is absent, wrong, uncertified, or exceeded.

In mathematical AI, the form is clear. A model replaces a theorem with a similar theorem whose hypotheses differ. It treats a group homomorphism as an isomorphism. It treats equivalence of categories as strict equality. It applies a compactness argument without compactness. It quotes a theorem that does not exist. It formalizes a statement that compiles but does not preserve the intended informal theorem. Each error is an unauthorized substitution. The output Y is presented as if it were admissible relative to X, but the certificate is missing or invalid.

This reframing is useful because it makes hallucination structurally preventable. The remedy is not merely to punish false outputs after generation. The remedy is to require the system to expose the admissibility policy before asserting the transformation. What kind of sameness is being claimed? What invariant must be preserved? What certificate is required? What obstruction would block the move? If the system cannot answer, it should mark the output as uncertified, speculative, heuristic, or inadmissible.

In mathematics, hallucination often arises from policy substitution. The model silently replaces a strong policy with a weak one. It uses similarity where equivalence is needed, cardinality where isomorphism is needed, homology where homotopy equivalence is needed, compilation where semantic fidelity is needed, or explanation where proof is needed. The surface output may look mathematically competent because a weaker relation really exists. But the task required a stronger policy. Hallucination is therefore not always pure invention. It is often wrong-level identity.

This diagnosis generalizes naturally. In scientific AI, hallucination may be treating a hypothesis as evidence, a correlation as causation, an animal result as a human result, or a preprint claim as settled fact. In legal AI, it may be treating a plausible citation as a real case, a general principle as binding law, or one jurisdiction as another. In memory systems, it may be treating inferred preference as remembered fact. In tool-using AI, it may be treating user intent as authorization for an action never approved. These are policy failures because the transformation exceeds the governing identity rule.

A future AI should therefore attach policy metadata to outputs. It should know whether it is operating under a factual policy, creative policy, mathematical proof policy, semantic summary policy, memory policy, citation policy, or tool-action policy. It should know when a generated sentence is a certified claim, a plausible hypothesis, a creative invention, or an uncertified analogy. Most dangerous hallucinations arise when these categories are collapsed.

The mathematical admissibility system provides the clean prototype. A theorem suggestion without proof is not false by default, but it is uncertified. A retrieved lemma is not applicable

until its hypotheses match. A formalization is not faithful until the translation policy is checked. A quotient is not lawful until well-definedness is certified. The system should learn to label these states.

Thus hallucination should be treated not as a mysterious defect of language models, but as a failure of policy-indexed identity. The AI said “same” without authority. Admissibility Algebra supplies the missing authority structure.

12.3 Abstraction as certified quotienting

Abstraction is the central operation of intelligence. To abstract is to ignore some distinctions and preserve others. Mathematics does this through quotienting, isomorphism classes, equivalence classes, homotopy types, moduli spaces, normal forms, universal properties, and categorical constructions. AI does it through embeddings, clustering, summarization, classification, representation learning, compression, and retrieval. The question is not whether abstraction should occur. It must. The question is whether the abstraction is certified.

Admissibility Algebra treats abstraction as quotient formation. A domain Ω is collapsed under policy π to form Ω/π. This quotient is useful only if π preserves the invariants required for the task. If π erases essential distinctions, the abstraction becomes destructive collapse. Thus future AI must distinguish certified quotienting from heuristic compression.

In mathematical AI, certified quotienting means that the system knows what equivalence relation, congruence, isomorphism class, homotopy relation, or proof equivalence is being used. It knows whether operations descend. It knows whether representatives can be replaced. It knows whether witness data must be retained. It knows whether the quotient is ordinary, categorical, homotopical, stack-like, or merely heuristic. This is quotient discipline.

Current AI systems often abstract by vector similarity. This is powerful for retrieval, but it is not a mathematical certificate. A vector embedding may place two theorem statements near each other because they share words, notation, topic, or proof style. But proximity does not establish that one theorem applies to the other. It does not preserve hypotheses, types, definitions, or proof obligations. Therefore vector abstraction should be labeled heuristic. It proposes candidates; it does not certify quotient identity.

Certified quotienting also matters for summaries. A proof summary is a quotient of the full proof. It forgets details. The question is whether it preserves the proof’s logical skeleton, dependencies, assumptions, and conclusion. A good summary may omit routine algebra but must not omit the only hypothesis that makes the theorem true. Thus summarization is an admissibility problem: what may be forgotten under the summary policy?

Formalization is another quotient-like transformation. Informal mathematical language contains ambiguity, convention, suppressed assumptions, and flexible notation. Formalization maps that cloud into precise declarations. The formal object is not literally the informal object. It is an admissible representation only if the formalization policy preserves the intended theorem. Thus formalization requires certified abstraction from informal variation to formal structure.

The same principle applies beyond mathematics. Scientific modeling is quotienting the physical world by assumptions and variables. Legal reasoning quotients events into legally relevant categories. Memory systems quotient experience into stored summaries. Planning systems quotient possible futures into action-relevant states. In every case, AI trust depends on whether the quotient preserves the invariants that matter.

Mathematics teaches this through exact examples. A quotient by a normal subgroup preserves group structure. A quotient by a non-normal subgroup may fail. A homotopy quotient preserves data lost by a naive orbit quotient. A stack preserves automorphism data lost by coarse moduli. These are not isolated technicalities. They are lessons for AI abstraction. If witness data matter, do not erase them. If operations must descend, prove well-definedness. If substitution depends on the quotient, record the policy.

Future AI should therefore include a quotient manager. It should track what domain is being collapsed, under which policy, with which certificate, preserving which invariants, and losing which distinctions. The slogan is simple: abstraction is safe only when quotienting is admissible.

12.4 Memory and reasoning as admissible continuity

A future AI will not merely answer isolated prompts. It will maintain memory, projects, goals, proofs, user preferences, evolving corpora, tool histories, and long chains of reasoning. This raises a new identity problem: what counts as the same object over time? A memory update, theorem refinement, proof repair, database merge, model revision, or plan adjustment is a transformation from an earlier state to a later state. Such transformations must preserve admissible continuity.

In mathematical AI, continuity over time appears when a theorem statement is revised, a definition is generalized, a proof is repaired, a formalization changes library versions, or a database object receives a new certificate. The system must know whether the new object replaces the old one, refines it, weakens it, contradicts it, or belongs to a different policy. A theorem with a missing hypothesis added is not the same theorem under all policies. A definition changed for convenience may alter downstream proofs. A proof script updated for a new library version may preserve theorem identity but change proof trace identity. These distinctions matter.

Admissible continuity requires an identity policy over time. Let S_t be the system state at time t, and S_{t+1} the updated state. The system should not merely overwrite S_t with S_{t+1}. It should record the transformation, policy, certificate, and preserved invariants. Did the update preserve theorem meaning? Did it preserve proof validity? Did it preserve user intention? Did it preserve source provenance? Did it preserve database references? Did it preserve quotient membership? Without such records, the system develops identity drift.

In mathematical work, identity drift can corrupt a corpus. A theorem name may come to point to a slightly different statement. A proof may be updated without recording changed assumptions. A formal declaration may be redefined in a way that invalidates earlier equivalences. A database merge may identify two nonidentical objects. A model may remember a conjecture as a theorem. Each case is a failure of admissible continuity.

The same problem appears in AI memory generally. A system may store a user's transient remark as a permanent preference, infer a belief that was never stated, merge two projects incorrectly, or remember a speculative idea as a settled conclusion. These are memory hallucinations. Under Admissibility Algebra, they are unauthorized transformations of identity across time. The remedy is the same: memory updates require policies and certificates.

Reasoning also requires continuity. A multi-step argument is a path of transformations. Each step must preserve the appropriate invariant. The final conclusion is trustworthy only if the chain is admissible. A single unauthorized substitution can corrupt the whole path. Thus long-horizon reasoning requires more than local fluency; it requires a stable admissibility trace. The system should be able to replay why each transformation was permitted.

This suggests a Lyapunov-style interpretation. Let $V\pi$ measure policy-relative tension: contradiction, invariant loss, uncertified substitution, identity drift, or obstruction accumulation. A stable admissibility process should not allow $V\pi$ to grow unchecked. Updates should reduce or control policy-relative tension by adding certificates, resolving obstructions, clarifying contexts, or weakening claims. Emergence of a stable mathematical object in the AI corpus occurs when transformations settle into a certified admissibility class.

The practical requirement is versioned admissibility. Every object, theorem, proof, certificate, quotient, and policy should have provenance and update history. The system should know whether a new version is equal, equivalent, refined, generalized, specialized, corrected, superseded, or incompatible with an old version. This is not clerical bookkeeping. It is identity preservation.

Thus memory and reasoning are not separate from admissibility. They are admissibility extended over time. A future AI becomes trustworthy when it knows not only what may count

as the same now, but what may continue to count as the same after transformation, update, compression, correction, and use.

12.5 The future AI slogan: intelligence is admissible transformation

The central slogan of this paper is that intelligence is admissible transformation. Current AI systems are powerful transformation engines. They transform prompts into answers, documents into summaries, languages into translations, informal claims into formal code, theorem states into tactic suggestions, examples into patterns, and text into embeddings. But transformation alone is not intelligence in the mathematical sense. It becomes intelligence when the system knows which transformations are permitted, under which policies, preserving which invariants, with which certificates.

This slogan deliberately shifts attention away from mere generation. A system that can generate plausible mathematical text is useful but incomplete. A system that can certify admissible transformations is qualitatively different. It knows when one object may replace another, when a theorem applies, when a quotient is lawful, when a proof step is valid, when an abstraction is safe, when an invariant blocks identification, and when the correct answer is undecided. That is mathematical judgment.

The same slogan also reframes AI safety. Many unsafe or untrustworthy behaviors can be understood as inadmissible transformations. Hallucination transforms uncertainty into fact. Sycophancy transforms user desire into truth. Bad memory transforms inference into remembered identity. Bad tool use transforms interpretation into unauthorized action. Bad abstraction transforms distinct people, cases, facts, or objects into a harmful quotient. Bad planning transforms a goal into a superficially similar but misaligned outcome. In each case, the system fails to preserve the invariant that should have governed the transformation.

Mathematics supplies the clean version of the problem. A theorem may be transformed into a summary only if theorem content is preserved. A proof sketch may be transformed into a formal proof only if proof obligations are met. A structure may be replaced by another only under the correct policy. A quotient may be formed only if operations and invariants descend. A retrieved lemma may be used only if its hypotheses match. This is the discipline that future AI must learn before it acts in messier domains.

The architecture follows the slogan. Beneath the admissibility layer, neural models may propose transformations. They may search, retrieve, analogize, compress, and generate. Above them, the admissibility layer selects policies, checks invariants, demands certificates, records obstructions, manages quotients, and limits substitution. The neural model supplies candidate motion. The admissibility system supplies lawful identity.

This architecture does not require eliminating creativity. On the contrary, it protects creativity by separating proposal from certification. The AI may freely generate conjectures, analogies, speculative mappings, possible proofs, and heuristic abstractions, provided it labels them correctly. What it may not do is present uncertified transformation as established identity. The boundary between exploration and assertion is the boundary of admissibility.

The research program is therefore staged. First, build Admissibility Algebra in mathematics. Second, construct an admissibility database of objects, policies, invariants, certificates, obstructions, quotients, and judgments. Third, train mathematical AI on certified transformations rather than answers alone. Fourth, connect the system to proof assistants as certificate checkers. Fifth, generalize the architecture to scientific reasoning, memory, language, law, and action. Mathematics is the seed crystal.

The long-term implication is that future AI should not be measured only by how much it can say or how fast it can generate. It should be measured by the transformations it can certify. A small system with explicit admissibility may be more trustworthy than a larger system with uncontrolled fluency. A model that knows when not to substitute may be more intelligent than a model that always completes the pattern.

The conclusion of the paper is therefore direct. Mathematical AI needs Admissibility Algebra because mathematics is not merely symbol generation. It is lawful transformation under identity constraints. The primitive judgment $\Gamma \vdash X \equiv_\pi Y : C$ captures the missing layer: context, policy, equivalence, and certificate. Quotients become controlled collapse. Hallucination becomes certificate failure. Proof assistants become admissibility engines. Training becomes learning what may count as the same.

The final slogan is the future AI thesis in one sentence: current AI predicts transformations; future AI must certify admissible transformations.

13. References

1. Youvan, D. C. Admissibility Algebra: Identity Policies, Invariant-Preserving Transformation, and a New Abstract Algebra Above Category Theory and Homotopy Type Theory. ResearchGate preprint, July 2026. DOI: 10.13140/RG.2.2.18444.07044.
2. Youvan, D. C. Reality Translates Contexts: From the Genetic Code Lookup Table to an Admissibility Theory of Physics, Mind, and Meaning. ResearchGate preprint, July 2026. DOI: 10.13140/RG.2.2.28090.96967.
3. Youvan, D. C. The Genetic Code as an Admissibility Algebra: Translation Quotients, Rank-10 Physicochemical Projection, and the G-A-C-U Topological Policy. ResearchGate preprint, July 2026. DOI: 10.13140/RG.2.2.35221.28643.
4. Youvan, D. C. Admissibility Fibrations: Policy-Indexed Identity, Certified Quotients, and Lyapunov-Stable Emergence. Manuscript, 2026.
5. Youvan, D. C. A Categorical-HoTT Formalization of Wigner's Friend as Contextual Gluing. ResearchGate preprint, July 2026. DOI: 10.13140/RG.2.2.35662.01607.
6. Youvan, D. C. Wigner's Friend as a Gluing Problem: Category Theory, Homotopy Type Theory, and the Relational Structure of Quantum Reality. ResearchGate preprint, July 2026. DOI: 10.13140/RG.2.2.23917.96487.
7. Youvan, D. C. Reality as Equivalence: Univalent Relational Dynamics, Category Theory, Homotopy Type Theory, and an Equivalence-First Foundation for Mathematical Physics. ResearchGate preprint, June 2026. DOI: 10.13140/RG.2.2.20432.26881.
8. Youvan, D. C. Equivalence-First Fractal Arithmetic: HoTT, Mandelbrot Symmetry, Riemann Zeta Duality, and the Prime Fixed-Locus Problem. ResearchGate preprint, June 2026. DOI: 10.13140/RG.2.2.29131.35363.
9. Bourbaki, N. Algebra I: Chapters 1–3. Springer, 1989.
10. Bourbaki, N. Theory of Sets. Springer, 2004.
11. Lang, S. Algebra. Revised third edition. Springer, 2002.
12. Dummit, D. S., and Foote, R. M. Abstract Algebra. Third edition. Wiley, 2004.
13. Mac Lane, S. Categories for the Working Mathematician. Second edition. Springer, 1998.
14. Awodey, S. Category Theory. Second edition. Oxford University Press, 2010.
15. Riehl, E. Category Theory in Context. Dover, 2016.

16. Leinster, T. Basic Category Theory. Cambridge University Press, 2014.

17. Borceux, F. Handbook of Categorical Algebra. Volumes 1–3. Cambridge University Press, 1994.

18. Johnstone, P. T. Sketches of an Elephant: A Topos Theory Compendium. Volumes 1–2. Oxford University Press, 2002.

19. Grothendieck, A. Revêtements étales et groupe fondamental. Séminaire de Géométrie Algébrique du Bois Marie 1960–1961, SGA 1. Springer Lecture Notes in Mathematics 224, 1971.

20. Grothendieck, A. Pursuing Stacks. Manuscript, 1983.

21. Giraud, J. Cohomologie non abélienne. Springer, 1971.

22. Lawvere, F. W., and Schanuel, S. H. Conceptual Mathematics: A First Introduction to Categories. Second edition. Cambridge University Press, 2009.

23. Lawvere, F. W. Functorial Semantics of Algebraic Theories. Proceedings of the National Academy of Sciences, 50, 869–872, 1963.

24. Eilenberg, S., and Mac Lane, S. General Theory of Natural Equivalences. Transactions of the American Mathematical Society, 58, 231–294, 1945.

25. Mac Lane, S. Homology. Springer, 1963.

26. Hatcher, A. Algebraic Topology. Cambridge University Press, 2002.

27. May, J. P. A Concise Course in Algebraic Topology. University of Chicago Press, 1999.

28. Spanier, E. H. Algebraic Topology. McGraw-Hill, 1966.

29. Whitehead, G. W. Elements of Homotopy Theory. Springer, 1978.

30. Quillen, D. Homotopical Algebra. Springer Lecture Notes in Mathematics 43, 1967.

31. Goerss, P. G., and Jardine, J. F. Simplicial Homotopy Theory. Birkhäuser, 1999.

32. Hovey, M. Model Categories. American Mathematical Society, 1999.

33. Hirschhorn, P. S. Model Categories and Their Localizations. American Mathematical Society, 2003.

34. Lurie, J. Higher Topos Theory. Princeton University Press, 2009.

35. Lurie, J. Higher Algebra. Manuscript, 2017.

36. Univalent Foundations Program. Homotopy Type Theory: Univalent Foundations of Mathematics. Institute for Advanced Study, 2013.

37. Martin-Löf, P. Intuitionistic Type Theory. Bibliopolis, 1984.

38. HoTT Book Authors. Homotopy Type Theory: Univalent Foundations of Mathematics. Institute for Advanced Study, 2013.

39. Voevodsky, V. Univalent Foundations Project. Institute for Advanced Study lectures and manuscripts, 2010–2014.

40. Awodey, S., and Warren, M. A. Homotopy Theoretic Models of Identity Types. Mathematical Proceedings of the Cambridge Philosophical Society, 146, 45–55, 2009.

41. Hofmann, M., and Streicher, T. The Groupoid Interpretation of Type Theory. In Twenty-Five Years of Constructive Type Theory, Oxford University Press, 1998.

42. Bezem, M., Coquand, T., and Huber, S. A Model of Type Theory in Cubical Sets. Leibniz International Proceedings in Informatics, 2014.

43. Cohen, C., Coquand, T., Huber, S., and Mörtberg, A. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. Leibniz International Proceedings in Informatics, 2018.

44. de Moura, L., Kong, S., Avigad, J., van Doorn, F., and von Raumer, J. The Lean Theorem Prover. Automated Deduction, CADE-25, Springer, 2015.

45. Avigad, J., de Moura, L., and Kong, S. Theorem Proving in Lean. Lean documentation project, 2015–2024.

46. The mathlib Community. The Lean Mathematical Library. Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, 2020.

47. Buzzard, K., Commelin, J., and Massot, P. Formalising Perfectoid Spaces. Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, 2020.

48. Paulson, L. C. Isabelle: A Generic Theorem Prover. Springer Lecture Notes in Computer Science 828, 1994.

49. Nipkow, T., Paulson, L. C., and Wenzel, M. Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Springer, 2002.

50. Bertot, Y., and Castéran, P. Interactive Theorem Proving and Program Development: Coq'Art. Springer, 2004.

51. Coq Development Team. The Coq Proof Assistant Reference Manual. Inria and contributors, continuing editions.

52. Bove, A., Dybjer, P., and Norell, U. A Brief Overview of Agda: A Functional Language with Dependent Types. Theorem Proving in Higher Order Logics, Springer, 2009.

53. Norell, U. Towards a Practical Programming Language Based on Dependent Type Theory. Ph.D. thesis, Chalmers University of Technology, 2007.

54. Harrison, J. Handbook of Practical Logic and Automated Reasoning. Cambridge University Press, 2009.

55. Nipkow, T., Klein, G. Concrete Semantics: With Isabelle/HOL. Springer, 2014.

56. Pierce, B. C. Types and Programming Languages. MIT Press, 2002.

57. Harper, R. Practical Foundations for Programming Languages. Second edition. Cambridge University Press, 2016.

58. Girard, J.-Y., Taylor, P., and Lafont, Y. Proofs and Types. Cambridge University Press, 1989.

59. Troelstra, A. S., and Schwichtenberg, H. Basic Proof Theory. Second edition. Cambridge University Press, 2000.

60. Baader, F., and Nipkow, T. Term Rewriting and All That. Cambridge University Press, 1998.

61. Dershowitz, N., and Jouannaud, J.-P. Rewrite Systems. In Handbook of Theoretical Computer Science, Volume B, Elsevier, 1990.

62. Huet, G. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. Journal of the ACM, 27, 797–821, 1980.

63. Barendregt, H. The Lambda Calculus: Its Syntax and Semantics. Revised edition. North-Holland, 1984.

64. Church, A. A Formulation of the Simple Theory of Types. Journal of Symbolic Logic, 5, 56–68, 1940.

65. Curry, H. B., and Feys, R. Combinatory Logic, Volume I. North-Holland, 1958.

66. Howard, W. A. The Formulae-as-Types Notion of Construction. In To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, 1980.

67. Curry, H. B. Functionality in Combinatory Logic. Proceedings of the National Academy of Sciences, 20, 584–590, 1934.

68. Gentzen, G. Investigations into Logical Deduction. Mathematische Zeitschrift, 39, 176–210 and 405–431, 1935.

69. Tarski, A. On the Concept of Logical Consequence. In Logic, Semantics, Metamathematics, Oxford University Press, 1956.

70. Gödel, K. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte für Mathematik und Physik, 38, 173–198, 1931.

71. Turing, A. M. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 42, 230–265, 1936.

72. Church, A. An Unsolvable Problem of Elementary Number Theory. American Journal of Mathematics, 58, 345–363, 1936.

73. Robinson, J. A. A Machine-Oriented Logic Based on the Resolution Principle. Journal of the ACM, 12, 23–41, 1965.

74. Davis, M., Logemann, G., and Loveland, D. A Machine Program for Theorem-Proving. Communications of the ACM, 5, 394–397, 1962.

75. Newell, A., Shaw, J. C., and Simon, H. A. Empirical Explorations with the Logic Theory Machine. Proceedings of the Western Joint Computer Conference, 1957.

76. Bundy, A. The Computer Modelling of Mathematical Reasoning. Academic Press, 1983.

77. Wos, L., Overbeek, R., Lusk, E., and Boyle, J. Automated Reasoning: Introduction and Applications. McGraw-Hill, 1992.

78. Blanchette, J. C., Greenaway, D., Kaliszyk, C., Kühlwein, D., and Urban, J. A Learning-Based Fact Selector for Isabelle/HOL. Journal of Automated Reasoning, 57, 219–244, 2016.

79. Kaliszyk, C., and Urban, J. Learning-Assisted Automated Reasoning with Flyspeck. Journal of Automated Reasoning, 53, 173–213, 2014.

80. Urban, J., and Jakubův, J. First Neural Conjecturing Datasets and Experiments. Intelligent Computer Mathematics, 2020.

81. Polu, S., and Sutskever, I. Generative Language Modeling for Automated Theorem Proving. arXiv preprint, 2020.

82. Han, J. M., Rute, J., Wu, Y., Ayers, E. W., and Polu, S. Proof Artifact Co-training for Theorem Proving with Language Models. International Conference on Learning Representations, 2022.

83. Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., and Anandkumar, A. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. Advances in Neural Information Processing Systems, 2023.

84. Zheng, K., Han, J. M., and Polu, S. MiniF2F: A Cross-System Benchmark for Formal Olympiad-Level Mathematics. International Conference on Learning Representations Workshop, 2022.

85. Polu, S., Han, J. M., Zheng, K., Baksys, M., Babuschkin, I., and Sutskever, I. Formal Mathematics Statement Curriculum Learning. International Conference on Learning Representations, 2023.

86. Wang, Q., Kaliszyk, C., and Urban, J. First Experiments with Neural Translation of Informal to Formal Mathematics. Certified Programs and Proofs Workshops, 2018.

87. Rabe, M. N., and Szegedy, C. Towards the Automatic Mathematician. Automated Deduction, CADE, 2021.

88. Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., Lackenby, M., Williamson, G., Hassabis, D., and Kohli, P. Advancing Mathematics by Guiding Human Intuition with AI. Nature, 600, 70–74, 2021.

89. Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., and Fawzi, A. Mathematical Discoveries from Program Search with Large Language Models. Nature, 625, 468–475, 2024.

90. Silver, D., et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature, 529, 484–489, 2016.

91. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention Is All You Need. Advances in Neural Information Processing Systems, 2017.

92. Brown, T. B., et al. Language Models Are Few-Shot Learners. Advances in Neural Information Processing Systems, 2020.

93. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-Augmented Generation for

Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 2020.

94. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense Passage Retrieval for Open-Domain Question Answering. Empirical Methods in Natural Language Processing, 2020.

95. Goodfellow, I., Bengio, Y., and Courville, A. Deep Learning. MIT Press, 2016.

96. Russell, S., and Norvig, P. Artificial Intelligence: A Modern Approach. Fourth edition. Pearson, 2021.

97. Pearl, J. Causality: Models, Reasoning, and Inference. Second edition. Cambridge University Press, 2009.

98. Pearl, J., and Mackenzie, D. The Book of Why. Basic Books, 2018.

99. Halpern, J. Y. Actual Causality. MIT Press, 2016.

100. Barwise, J., and Seligman, J. Information Flow: The Logic of Distributed Systems. Cambridge University Press, 1997.

101. Gärdenfors, P. Conceptual Spaces: The Geometry of Thought. MIT Press, 2000.

102. Carnap, R. Meaning and Necessity. University of Chicago Press, 1947.

103. Quine, W. V. O. Word and Object. MIT Press, 1960.

104. Putnam, H. The Meaning of “Meaning.” Minnesota Studies in the Philosophy of Science, 7, 131–193, 1975.

105. Kripke, S. Naming and Necessity. Harvard University Press, 1980.

106. Wittgenstein, L. Philosophical Investigations. Blackwell, 1953.

107. Frege, G. Begriffsschrift. Halle, 1879.

108. Frege, G. On Sense and Reference. Zeitschrift für Philosophie und philosophische Kritik, 100, 25–50, 1892.

109. Noether, E. Invariante Variationsprobleme. Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse, 235–257, 1918.

110. Weyl, H. Symmetry. Princeton University Press, 1952.

111. Weyl, H. Space-Time-Matter. Dover reprint, 1952.

112. Yang, C. N., and Mills, R. L. Conservation of Isotopic Spin and Isotopic Gauge Invariance. Physical Review, 96, 191–195, 1954.

113. Utiyama, R. Invariant Theoretical Interpretation of Interaction. Physical Review, 101, 1597–1607, 1956.

114. Lyapunov, A. M. The General Problem of the Stability of Motion. 1892. English translation, Taylor & Francis, 1992.

115. Strogatz, S. H. Nonlinear Dynamics and Chaos. Second edition. Westview Press, 2015.

116. Arnold, V. I. Ordinary Differential Equations. Springer, 1992.

117. Hirsch, M. W., Smale, S., and Devaney, R. L. Differential Equations, Dynamical Systems, and an Introduction to Chaos. Third edition. Academic Press, 2013.

118. Gelfand, I. M., and Manin, Y. I. Methods of Homological Algebra. Second edition. Springer, 2003.

119. Weibel, C. A. An Introduction to Homological Algebra. Cambridge University Press, 1994.

120. Kashiwara, M., and Schapira, P. Categories and Sheaves. Springer, 2006.

The author has published over 5,600 AI-assisted papers at: https://www.researchgate.net/profile/Douglas-Youvan/research . Google Scholar has indexed these preprints, and if you want a particular reference, the AI mode of a Google search (Gemini) has efficiently catalogued these preprints.

Appendix A. Minimal Schema

A.1 Purpose of the Schema

The minimal schema is the implementation-facing skeleton of Admissibility Algebra for Mathematical AI. Its purpose is not to encode all of mathematics at once. Its purpose is to define the smallest durable set of records needed to store policy-indexed identity, certified transformation, controlled quotienting, scoped substitution, and proof-grounded failure. The schema is therefore deliberately modest. It begins with objects, policies, invariants, certificates, obstructions, quotients, judgments, sources, formal systems, proof traces, model outputs, and verification status.

The central design rule is that the database should not store mathematical claims as free-floating text. It should store admissibility events. An admissibility event occurs when some object X is compared with some object Y in context Γ under policy π, producing either a certificate C, an obstruction O, an undecided state, or an uncertified proposal. This record is the atomic memory of the system.

A conventional theorem database asks: what theorem is this? A vector database asks: what text is nearby? A proof assistant asks: does this proof check? An admissibility database asks: under what policy may this object stand in for that one, preserving what invariants, with what certificate, and in which substitution scope? The schema must therefore preserve the difference between resemblance, relevance, equivalence, proof, obstruction, and permission.

The minimal schema should support five core operations. First, it should store mathematical objects without prematurely identifying their presentations. Second, it should store policies as first-class entities. Third, it should link policies to invariants, certificates, and substitution scopes. Fourth, it should store positive and negative judgments. Fifth, it should connect all records to sources, formal systems, proof traces, and verification status.

The schema should also be versioned. Mathematical libraries evolve. Definitions change. Formal systems update. Proof scripts break. Papers are revised. A judgment valid in one context may become invalid or require repair in another. Therefore each record should have provenance, timestamps, source identifiers, and formal-system metadata. This prevents the database from confusing historical, current, formal, informal, and speculative records.

The schema is minimal, but it should not be simplistic. The first prototype may use tables and text fields, but the conceptual schema already assumes a relational layer, graph layer, and vector layer. The relational layer stores records. The graph layer stores typed relationships. The vector layer retrieves semantically related objects. The proof assistant verifies certificates. These layers should remain conceptually distinct even if the first implementation is small.

A.2 MathematicalObject

A MathematicalObject is any entity that can enter an admissibility judgment. It may be a term, theorem, proof, definition, group, ring, module, category, type, topological space, diagram, quotient, formal declaration, informal statement, algorithm, tactic state, or representation. The schema should not restrict objects too narrowly at first because mathematical AI must compare across representational levels.

Minimal fields:

ObjectID
Name
ObjectKind
Domain
FormalSystem
FormalEncoding
InformalDescription
SourceID
Namespace
ContextID
Version
Hash
CreatedAt
UpdatedAt
EmbeddingID
Status

ObjectKind distinguishes theorem, definition, structure, proof, term, type, category, group, space, diagram, algorithm, proof state, and other broad classes. Domain records algebra, topology, category theory, HoTT, logic, proof assistant, analysis, geometry, computation, or cross-domain. FormalSystem identifies Lean, Coq, Agda, Isabelle, informal prose, Python, symbolic computation, or none. FormalEncoding stores the machine-readable object when available. InformalDescription stores human-readable text.

A MathematicalObject should not assume uniqueness of name. The same name may occur in different libraries, contexts, notational traditions, or papers. Therefore Name is not identity. ObjectID is the database identity. FormalEncoding and SourceID help disambiguate. Hash may identify exact text, formal code, or serialized object content.

Objects may be linked by presentation relationships. One object may be a formalization of another, a normal form of another, a representative of a quotient class, a generated version of

another, a corrected version of another, or an explanation of another. These links should not be stored as untyped “related_to” edges. They should be policy-aware relationships mediated by admissibility judgments.

A first database may store informal objects and formal objects side by side. For example, an informal theorem statement from a paper and a Lean theorem declaration are both MathematicalObjects. Their relationship is not equality. It is an admissibility claim under a formalization-fidelity policy. The schema must allow that relationship to be certified, rejected, or left undecided.

MathematicalObject records should be immutable where possible, with revisions stored as new versions. If an object is edited, the previous version should remain addressable. This allows the system to record whether a new version is equal, equivalent, corrected, generalized, specialized, or incompatible with the old one. Versioned objects are essential for admissible memory over time.

A.3 Policy

A Policy is a first-class rule of permitted sameness. It defines the conditions under which X and Y may count as identical, equivalent, substitutable, comparable, quotientable, or transportable for a specified purpose. Policies are the central innovation of the schema. They prevent the database from collapsing all relations into generic similarity.

Minimal fields:

PolicyID
Name
PolicyKind
Domain
FormalDefinition
InformalDescription
CertificateTypeID
InvariantSetID
SubstitutionScopeID
StrengthLevel
ParentPolicies
ChildPolicies
CompatiblePolicies
IncompatiblePolicies
CreatedAt

UpdatedAt
Status

PolicyKind may include equality, definitional equality, propositional equality, isomorphism, equivalence, homotopy equivalence, quotient equivalence, setoid equivalence, logical equivalence, semantic equivalence, approximation, implication, refinement, specialization, formalization fidelity, proof equivalence, or heuristic similarity.

A Policy must specify what kind of certificate it accepts. Group isomorphism accepts structure-preserving inverse maps. Definitional equality accepts reduction. Propositional equality accepts proof terms. Homotopy equivalence accepts maps plus homotopies. Formalization fidelity may accept a combination of expert annotation, theorem alignment, examples, and formal proof support. Similarity may accept embeddings, but should be marked heuristic and non-certifying unless strengthened by other evidence.

Policies should be arranged in a policy graph. Equality may imply isomorphism in some contexts. Homeomorphism may imply homotopy equivalence. Diffeomorphism may imply homeomorphism. Ordered-ring isomorphism may imply ring isomorphism. However, policies are not always linearly ordered. Some are incomparable. The schema should therefore allow multiple parent and child policies rather than a single hierarchy.

The database should distinguish certified policies from heuristic policies. A heuristic similarity policy may be useful for retrieval but should not license proof substitution. A proof-equivalence policy may be formal and strong. A semantic-preservation policy may be partially formal, partly human-annotated. This distinction prevents the vector layer from becoming an unauthorized authority.

A Policy should also specify failure modes. Under group isomorphism, element-order mismatch may obstruct. Under quotient algebra formation, non-congruence may obstruct. Under theorem application, missing hypotheses may obstruct. Under formalization fidelity, altered quantifier structure may obstruct. Storing failure modes helps the AI diagnose inadmissibility instead of merely reporting failure.

A.4 Invariant

An Invariant is something that must be preserved under a policy for the admissibility judgment to hold. Invariants are the content of policy. They state what must not change.

Minimal fields:

InvariantID
Name
InvariantKind

Domain
Definition
FormalPredicate
InformalDescription
PreservedByPolicies
CompleteForClasses
ObstructionPower
ComputationMethod
SourceID
CreatedAt
UpdatedAt
Status

InvariantKind may include algebraic, topological, categorical, homotopical, logical, type-theoretic, computational, semantic, syntactic, proof-theoretic, or heuristic. Algebraic examples include order, rank, kernel, image, spectrum, dimension, characteristic, element-order spectrum, commutativity, nilpotence, and ideal structure. Topological examples include compactness, connectedness, Hausdorff property, fundamental group, homology, and homotopy type. Categorical examples include universal properties, adjunctions, naturality, limits, colimits, and equivalence-invariant properties.

The field PreservedByPolicies records which policies preserve the invariant. For example, compactness is preserved by homeomorphism but not by arbitrary homotopy equivalence. Homology is preserved by homotopy equivalence. Group element-order structure is preserved by group isomorphism. Type correctness is preserved by valid formal transformations. This allows the system to ask whether a property may be transported under π.

CompleteForClasses records whether the invariant classifies objects in a specified domain. Dimension classifies finite-dimensional vector spaces over a fixed field up to isomorphism. Cardinality classifies finite sets up to bijection. Cardinality does not classify finite groups up to isomorphism. Homology does not generally classify spaces up to homotopy equivalence. This field prevents the AI from treating matching invariants as complete certificates when they are only suggestive.

ObstructionPower records whether mismatch in the invariant decisively blocks a policy. If an invariant preserved by π differs between X and Y, then $X \equiv_\pi Y$ is impossible. This allows rapid negative judgments. For example, if two groups have different orders, group isomorphism is obstructed. If two spaces have different homology groups, homotopy equivalence is obstructed. If two theorem statements have different quantifier structure under a strict semantic policy, formalization fidelity may be obstructed.

ComputationMethod records how the invariant can be computed, proved, retrieved, or estimated. Some invariants are computed directly. Some require proof. Some require external tools. Some are undecidable or expensive. The database should not assume all invariants are cheaply available.

An Invariant may be attached to many policies. It may also appear in certificates and obstructions. The same invariant can function positively as preserved content and negatively as an obstruction when mismatched. Therefore invariants should be indexed as reusable mathematical objects, not buried inside explanatory text.

A.5 Certificate

A Certificate is the witness that an admissibility judgment holds. It is the positive evidence for Γ ⊢ X ≡π Y : C.

Minimal fields:

CertificateID
CertificateKind
PolicyID
ObjectXID
ObjectYID
ContextID
FormalSystem
CertificateContent
Checker
VerificationStatusID
ProofTraceID
SourceID
Canonicality
Constructivity
Dependencies
CreatedAt
UpdatedAt
Status

CertificateKind may include proof object, proof term, theorem reference, tactic proof, explicit isomorphism, equivalence data, homotopy data, natural isomorphism, quotient relation witness, computation trace, normalization result, semantic alignment, human annotation, or external verification.

CertificateContent stores the witness itself or a pointer to it. In formal settings, this may be a proof term, theorem name, Lean declaration, Coq term, Isabelle proof, Agda construction, or serialized artifact. In informal settings, it may be a structured proof sketch or human annotation, but the VerificationStatus must reflect that it is not kernel-checked.

Checker records the authority that validated the certificate. It may be Lean kernel, Coq kernel, Isabelle, Agda, Python test, symbolic computation, human expert, or none. VerificationStatus distinguishes verified, failed, unchecked, partially checked, heuristic, obsolete, or superseded.

Canonicality records whether the certificate is canonical, noncanonical, choice-dependent, arbitrary, or unknown. This matters because noncanonical certificates may not support stable substitution or database normalization. For example, an isomorphism from a finite-dimensional vector space to k^n depends on a basis unless a canonical basis is given.

Constructivity records whether the certificate gives explicit data or merely proves existence. A nonconstructive proof that an isomorphism exists may be admissible for some classification claims but insufficient for computation, transport, or implementation. This field helps the system choose certificate strength appropriate to the task.

Dependencies record the definitions, theorems, libraries, assumptions, and previous certificates used. A certificate without dependencies cannot be audited. If a dependency changes or is invalidated, the certificate may require rechecking. Thus the certificate record is part of the versioned mathematical memory.

A.6 Obstruction

An Obstruction is a certified reason why an admissibility judgment fails under a specified policy. It is the negative counterpart of a certificate.

Minimal fields:

ObstructionID
ObstructionKind
PolicyID
ObjectXID
ObjectYID
ContextID
ViolatedInvariantID
ObstructionContent
ProofOrEvidence
Checker
VerificationStatusID

SourceID
Dependencies
CreatedAt
UpdatedAt
Status

ObstructionKind may include invariant mismatch, counterexample, missing hypothesis, type mismatch, failed transport, non-congruence, failed naturality, failed continuity, failed invertibility, failed well-definedness, failed formalization fidelity, failed theorem application, or kernel rejection.

ViolatedInvariantID identifies the invariant whose failure blocks admissibility. For example, element-order mismatch blocks group isomorphism; compactness mismatch blocks homeomorphism; homology mismatch blocks homotopy equivalence; missing normality blocks quotient group formation; missing hypothesis blocks theorem application. When possible, the obstruction should be linked to a reusable invariant record.

ObstructionContent stores the specific mismatch or failure. For $Z/4Z$ and $Z/2Z \times Z/2Z$ under group isomorphism, the content may state that one contains an element of order 4 and the other does not. For quotient failure, it may state that the proposed equivalence relation is not compatible with multiplication. For formal proof failure, it may store the failed proof state and error message.

ProofOrEvidence stores the negative certificate. In mathematics, an obstruction should be more than “I could not find a proof.” It should supply a reason. If no formal obstruction exists, the status should be undecided or search_failed rather than inadmissible. This distinction is essential for mathematical honesty.

Checker and VerificationStatus work as in Certificate. A formal obstruction may be machine-checked. A human-identified obstruction may be annotated. A suspected obstruction may be heuristic. The database must avoid promoting unverified failure to proof of inadmissibility.

Obstructions are critical for learning. They prevent repeated invalid attempts. They teach policy boundaries. They allow the AI to answer with precision: not admissible under π because invariant I fails, but possibly admissible under weaker policy π'. Negative knowledge is therefore as valuable as positive knowledge.

A.7 Quotient

A Quotient records a controlled collapse Ω/π. It represents a domain after a policy has declared which distinctions no longer count.

Minimal fields:

QuotientID
BaseDomainID
PolicyID
QuotientObjectID
RelationDefinition
RepresentativePolicy
PreservedInvariantSetID
CollapsedDistinctions
RetainedWitnessData
WellDefinednessCertificateID
SubstitutionScopeID
SourceID
CreatedAt
UpdatedAt
Status

BaseDomainID identifies Ω. PolicyID identifies π. QuotientObjectID identifies Ω/π when constructed as a MathematicalObject. RelationDefinition states the equivalence relation, congruence, group action, homotopy relation, proof equivalence, semantic relation, or other collapse rule.

RepresentativePolicy records how representatives are chosen, if at all. Some quotients have canonical representatives. Others do not. Some require arbitrary choices. Some should retain multiple representatives. This matters because AI systems often hallucinate canonical representatives where none exist.

PreservedInvariantSetID identifies what survives the quotient. CollapsedDistinctions states what is intentionally forgotten. These fields make abstraction auditable. The system should know what it lost when forming Ω/π.

RetainedWitnessData records whether the quotient keeps certificates, automorphisms, paths, stabilizers, coherence data, or other higher structure. Ordinary set quotients may discard witnesses. Homotopy quotients, stacks, groupoids, and higher quotients may retain them. Mathematical AI should not assume all quotients are coarse equivalence classes.

WellDefinednessCertificateID is essential. A quotient by an arbitrary relation may not preserve operations. A lawful algebraic quotient requires compatibility. This field points to the certificate proving that the quotient construction is valid under the intended policy.

SubstitutionScopeID states what reasoning is valid in the quotient world. A quotient permits some substitutions and forbids others. The system must know whether it may compute, rewrite, transport properties, or only classify.

The Quotient record is the database form of abstraction. It prevents the AI from compressing mathematical structure without knowing what was preserved.

A.8 AdmissibilityJudgment

The AdmissibilityJudgment is the atomic record of the system. It stores the result of comparing X and Y under policy π in context Γ.

Minimal fields:

JudgmentID
ContextID
ObjectXID
ObjectYID
PolicyID
ResultStatus
CertificateID
ObstructionID
InvariantSetID
SubstitutionScopeID
Directionality
Confidence
VerificationStatusID
SourceID
CreatedAt
UpdatedAt
Version
Notes

ResultStatus includes admissible, inadmissible, undecided, uncertified, heuristic, obsolete, superseded, or inconsistent. CertificateID is filled when positive. ObstructionID is filled when negative. Both should not be simultaneously authoritative unless the record concerns different contexts or indicates inconsistency.

Directionality records whether the judgment is symmetric, directed, reversible, approximate, refinement, implication, specialization, generalization, quotient projection, embedding, or transport. Not all admissibility judgments are symmetric. A summary may be an admissible compression of a proof, but the proof cannot be reconstructed from the summary. A quotient

map is directed. An implication is directed. A formalization may be intended as a faithful representation but not a reversible transformation.

Confidence is not a substitute for proof. It is useful only for heuristic, uncertain, or model-generated records. A formally verified judgment may have confidence 1 by convention, but the real authority is VerificationStatus. A plausible model output may have high confidence but remain uncertified.

The judgment record should support multi-policy comparison. The same ObjectXID and ObjectYID may have many records under different PolicyIDs. The system should be able to retrieve an identity profile: same under cardinality, not same under group isomorphism, undecided under stronger policy, equivalent under quotient policy.

The judgment record is also the audit trail for substitution. If the AI uses Y in place of X, it should be able to cite the JudgmentID that licensed the move. This is how mathematical AI becomes traceable.

A.9 SourceDocument

A SourceDocument records the origin of objects, policies, certificates, or obstructions.

Minimal fields:

SourceID
Title
Authors
SourceType
PublicationDate
DOI
URL
FilePath
FormalLibrary
FormalSystem
Version
Hash
CitationText
Notes

SourceType may include paper, book, formal library, theorem prover file, database import, human annotation, model output, computation log, proof trace, or conversation. Hash allows exact identification of files or code. Version is essential for formal libraries because theorem names, definitions, and proofs can change.

The database should distinguish primary mathematical sources from generated or secondary sources. A model output can be a SourceDocument, but it should not be treated as authoritative unless certified. A formal library source may be authoritative within its formal system. A paper may contain informal mathematics requiring interpretation. Source provenance prevents silent mixing of authority levels.

A.10 FormalSystem

A FormalSystem records the formal environment in which objects or certificates are interpreted.

Minimal fields:

FormalSystemID
Name
Version
Logic
Kernel
LibraryVersion
PackageSet
ImportContext
TrustLevel
Notes

Name may be Lean, Coq, Agda, Isabelle, HOL Light, Metamath, informal mathematics, Python, symbolic algebra, or other. Logic records dependent type theory, CIC, HOL, set theory, cubical type theory, or another foundation. Kernel records the trusted checker when applicable.

FormalSystem is part of Γ. A proof certificate valid in Lean is valid within Lean's formal environment, not automatically in all foundations. Translation between formal systems is itself an admissibility problem requiring a policy and certificate.

A.11 ProofTrace

A ProofTrace records the sequence of transformations leading to a proof or failed proof attempt.

Minimal fields:

ProofTraceID
FormalSystemID
InitialGoal
FinalStatus
TraceContent

TacticSequence
IntermediateStates
Dependencies
FailureReason
CertificateID
ObstructionID
Runtime
CreatedAt
UpdatedAt

ProofTrace is useful for training because it shows proof-state movement. Successful traces teach admissible transformation. Failed traces teach obstruction, missing hypotheses, or invalid tactics. A trace may point to a Certificate if successful or an Obstruction if failure is diagnosed.

The database should preserve failed proof traces when possible. They are not noise. They are negative training data for admissibility.

A.12 ModelOutput

A ModelOutput records what an AI system generated before or after verification.

Minimal fields:

ModelOutputID
ModelName
ModelVersion
Prompt
OutputContent
TaskType
ProposedPolicyID
ProposedJudgmentID
VerificationStatusID
LinkedCertificateID
LinkedObstructionID
CreatedAt
Notes

This table prevents model outputs from being confused with verified mathematics. A generated theorem, proof sketch, formalization, or equivalence claim remains a ModelOutput until linked to an admissibility judgment with verification status. This allows AI creativity without contaminating the certified corpus.

A.13 VerificationStatus

VerificationStatus records the authority level of a certificate, obstruction, judgment, or output.

Minimal statuses:

verified_formal
verified_computational
verified_human
partially_verified
unchecked
failed
heuristic
obsolete
superseded
contradicted
undecided

VerificationStatus should be separate from Confidence. Confidence is model belief or heuristic strength. VerificationStatus is epistemic authority. A high-confidence generated proof is still unchecked until verified. A low-confidence formal proof that passes the kernel is verified within its context.

A.14 Minimal Working Tables

A minimal implementation can begin with the following tables:

MathematicalObject
Policy
Invariant
Certificate
Obstruction
Quotient
AdmissibilityJudgment
SourceDocument
FormalSystem
ProofTrace
ModelOutput
VerificationStatus

Additional link tables will be needed:

PolicyInvariant
PolicyRefinement
PolicyCertificateType
ObjectRelation
CertificateDependency
ObstructionInvariant
JudgmentInvariant
QuotientInvariant
ProofTraceStep
SourceObject
SourceJudgment

These link tables prevent overloaded fields and allow many-to-many structure.

A.15 Minimal Graph Edges

The graph layer should support typed edges:

OBJECT formalizes OBJECT
OBJECT presents OBJECT
OBJECT represents OBJECT
OBJECT normalizes_to OBJECT
OBJECT quotient_of OBJECT
OBJECT member_of QUOTIENT
POLICY refines POLICY
POLICY weakens POLICY
POLICY preserves INVARIANT
POLICY violates_possible INVARIANT
CERTIFICATE certifies JUDGMENT
OBSTRUCTION obstructs JUDGMENT
INVARIANT obstructs POLICY_WHEN_MISMATCHED
QUOTIENT collapses DISTINCTION
JUDGMENT licenses SUBSTITUTION_SCOPE
THEOREM depends_on THEOREM
PROOFTRACE produces CERTIFICATE
MODELOUTPUT proposes JUDGMENT
FORMALSYSTEM checks CERTIFICATE

These edges should be typed because generic relatedness is not enough. The entire point of Admissibility Algebra is to prevent undisciplined relation from masquerading as identity.

A.16 Minimal Prototype Goal

The minimal schema is sufficient if it can support the following demonstration:

Store Z/4Z as a MathematicalObject.
Store Z/2Z × Z/2Z as a MathematicalObject.
Store cardinality policy.
Store group-isomorphism policy.
Store cardinality invariant.
Store element-order invariant.
Store a positive judgment under cardinality.
Store a negative judgment under group isomorphism.
Store the obstruction: element-order mismatch.
Return a multi-policy answer.

The expected output is:

Z/4Z and Z/2Z × Z/2Z are admissibly the same under cardinality policy, with certificate that both have four elements. They are inadmissible under group-isomorphism policy, with obstruction that element-order structure differs: Z/4Z has an element of order 4, while Z/2Z × Z/2Z does not.

If the schema can represent this cleanly, it can represent the core idea. Everything else is scale.

Appendix B. Central Machine-Readable Record

B.1 Purpose of the Record

The central machine-readable record of Admissibility Algebra is the admissibility judgment:

$\Gamma \vdash X \equiv_\pi Y : C$

and its negative or unresolved variants:

$\Gamma \nvdash X \equiv_\pi Y$ because $O_\pi(X,Y)$

$\Gamma \ ?\vdash X \equiv_\pi Y$

$\Gamma \vdash ?\ X \equiv_\pi Y$

These forms distinguish certified admissibility, certified inadmissibility, undecided status, and uncertified proposal. The record is intended to be read both mathematically and computationally. Mathematically, it states the conditions under which one object may count as the same as another. Computationally, it defines the atomic unit stored, queried, checked, trained on, and reused by a mathematical AI system.

The purpose of the record is to prevent undisciplined substitution. A conventional AI output may say “X and Y are equivalent” without specifying what kind of equivalence is meant, what context is assumed, what invariant survives, or what certificate proves the claim. The admissibility record refuses that ambiguity. It forces every sameness claim into a structured form: context, objects, policy, certificate, invariants, substitution scope, and verification status.

The record should be machine-readable because mathematical AI must not rely on prose alone. Prose explanations are useful for humans, but proof-grounded systems need structured fields. The machine must know which component is the policy, which component is the certificate, which invariant was checked, which obstruction applies, and whether the result is verified or merely proposed. Without this structure, the AI cannot learn admissibility; it can only imitate mathematical language.

The record is also designed to support multiple levels of rigor. Some entries may be fully formal, checked by Lean or another proof assistant. Some may be computationally verified. Some may be human-annotated. Some may be heuristic proposals generated by a model. The record should not erase these distinctions. Verification status is part of the judgment. The same apparent claim can exist as an uncertified proposal, a partially verified argument, and finally a formal certificate.

The record is the bridge between paper theory and implementation. It turns the philosophical thesis “identity is policy-indexed” into a data object. Every database row, graph edge, proof trace, model output, and training example should ultimately connect to this record type.

B.2 Positive Record: Certified Admissibility

The positive record represents a certified judgment:

$\Gamma \vdash X \equiv_\pi Y : C$

This means that, in context Γ, X and Y count as the same under policy π, with certificate C. A positive record is not merely a similarity score or generated statement. It asserts that an admissibility certificate has been provided and validated according to the record’s verification standard.

A minimal positive record should contain:

JudgmentID
Context Γ
Object X
Object Y
Policy π
Certificate C
Invariant family I_π
Substitution scope S_π
Verification status
Source provenance
Dependency list
Version information
Timestamp

The context Γ defines the jurisdiction. It may include a formal system, theorem library, assumptions, definitions, universe levels, declared variables, domain restrictions, or informal mathematical setting. The objects X and Y are the compared entities. The policy π states the kind of sameness. The certificate C is the witness. The invariant family I_π states what was preserved. The substitution scope S_π states what the certificate permits the system to do.

For example:

Γ: finite group theory
X: $\mathbb{Z}/4\mathbb{Z}$
Y: cyclic group of order 4
π: group isomorphism

C: explicit isomorphism from residue class 1 to chosen generator
Iπ: group operation, identity, inverses, element order
Sπ: group-theoretic properties
Verification: human/formal/computational depending on implementation

This record permits the system to treat X and Y as the same for group-theoretic reasoning. It does not permit every possible substitution. If one object has a particular computational encoding or presentation-specific name, that extra data may not be preserved. The substitution scope prevents overuse of the certificate.

Another example:

Γ: finite-dimensional vector spaces over field k
X: vector space V with chosen basis b
Y: k^n
π: coordinate isomorphism relative to basis b
C: map sending v to coordinate vector in basis b
Iπ: linear structure
Sπ: linear-algebraic computation respecting the chosen basis
Verification: formal if encoded

Here the certificate is basis-dependent. The record should mark noncanonicality. If no basis is specified, an isomorphism may exist but not be computationally available. Thus the certificate field and canonicality metadata matter.

A positive record should also record whether the certificate composes with others. If X ≡π Y and Y ≡π Z are certified, the system may be able to construct X ≡π Z. But this depends on policy and coherence. The record should therefore support certificate composition or at least point to composition rules defined by π.

The positive record is what allows mathematical AI to use previous work safely. When it later substitutes Y for X, it should cite the positive record that licenses the move. If no such record exists, the substitution should remain uncertified.

B.3 Negative Record: Certified Inadmissibility

The negative record represents certified inadmissibility:

Γ ⊬ X ≡π Y because Oπ(X,Y)

This means that, in context Γ, X and Y do not count as the same under policy π, because obstruction O shows that a required invariant fails, a hypothesis is absent, a certificate type cannot be inhabited, or a transport is invalid.

A minimal negative record should contain:

JudgmentID
Context Γ
Object X
Object Y
Policy π
Obstruction O
Violated invariant
Proof or evidence of violation
Verification status
Source provenance
Dependency list
Version information
Timestamp

A negative record must not be created merely because proof search failed. Failure to find a certificate is not proof of inadmissibility. The negative record requires an obstruction. The obstruction may be a counterexample, invariant mismatch, type error, missing hypothesis, failed naturality condition, non-congruence, failed well-definedness proof, or formal rejection with diagnostic explanation.

Example:

Γ: finite group theory
X: Z/4Z
Y: Z/2Z × Z/2Z
π: group isomorphism
O: element-order invariant mismatch
Violated invariant: existence of element of order 4
Proof: Z/4Z has element [1] of order 4; all nonidentity elements of Z/2Z × Z/2Z have order 2
Verification: human/formal/computational depending on implementation

This record blocks any later attempt to treat X and Y as group-isomorphic. It also guides the system toward weaker policies. The same pair may still be positive under cardinality policy. The negative record does not say “these are unrelated”; it says “this policy fails.”

Another example:

Γ: quotient group construction
X: group G with subset S
Y: proposed quotient G/S

π: quotient group admissibility
O: S is not normal in G
Violated invariant: coset multiplication well-definedness
Proof: there exist g,s such that $gsg^{-1} \notin S$
Verification: formal or human

This record prevents destructive quotienting. It identifies exactly what must be repaired: replace S with a normal subgroup, pass to the normal closure, or weaken the intended structure.

A negative record is valuable training data. It teaches the AI how to refuse with mathematical content. A mature system should say not merely “no,” but “no under π, because invariant I fails; a weaker policy π′ may still hold.” This is the core of policy-indexed precision.

B.4 Undecided Record

The undecided record represents the state:

$\Gamma \not\vdash X \equiv_\pi Y$

This means that the system does not currently have either a certificate or an obstruction. The claim is open relative to the database, search procedure, formal system, or current mathematical knowledge. Undecided is not false. It is not admissible. It is a distinct epistemic state.

A minimal undecided record should contain:

JudgmentID
Context Γ
Object X
Object Y
Policy π
Search status
Attempted certificates
Attempted invariants
Attempted proof traces
Known partial evidence
Known partial obstructions
Verification status
Timestamp

Undecided records are important because mathematical AI must not be forced into binary classification. Many mathematical questions are hard. Some require deeper search. Some are open. Some are merely outside the current corpus. Some are undecided because the objects

are incompletely specified. The system must be allowed to say that admissibility has not been established.

Undecided records should preserve search history. If the system tried to construct an isomorphism and failed, the attempt should be recorded. If it computed several invariants and found them matching, that should be recorded as partial evidence. If it searched for obstructions and found none, that should be recorded without overstating the result. This prevents repeated work and supports later refinement.

Example:

Γ: topological spaces
X: space A
Y: space B
π: homeomorphism
Status: undecided
Partial evidence: same homology groups, both compact, both connected
Missing: explicit homeomorphism or obstruction
Warning: shared homology does not certify homeomorphism

This record teaches the model to avoid overclaiming. Similar invariants may support search but not certification.

Another example:

Γ: formalization fidelity
X: informal theorem from paper
Y: Lean declaration
π: semantic formalization fidelity
Status: undecided
Partial evidence: formal theorem proves analogous result
Missing: human annotation of suppressed assumptions; proof alignment; examples
Warning: compilation certifies formal theorem, not necessarily fidelity to X

Undecided records are useful precisely because they keep uncertainty structured. The model does not have to hallucinate closure.

B.5 Uncertified Proposal Record

The uncertified proposal record represents:

$\Gamma \vdash ?\ X \equiv_\pi Y$

This means that a model, user, retrieval system, heuristic search, or informal argument has proposed an admissibility claim, but no certificate or obstruction has yet been attached. This is the natural state of AI-generated mathematics before verification.

A minimal uncertified record should contain:

ProposalID
Proposed JudgmentID
Context Γ
Object X
Object Y
Proposed policy π
Proposed explanation
Proposed certificate type
Model or source
Confidence
Verification status: unchecked
Timestamp

The uncertified proposal record protects the database from contamination. It allows creativity and speculation without treating generated output as verified knowledge. A model may propose that two objects are isomorphic, that a theorem applies, that a quotient is lawful, or that an informal statement has been formalized. The system can store the proposal, but it must remain uncertified until checked.

This distinction is critical for AI workflows. Generative systems are useful because they propose. But their proposals must not be inserted into the certified corpus as facts. The proposal record acts as a quarantine layer. It permits review, proof search, human annotation, and formal verification before promotion.

Example:

ModelOutput: “These two groups are isomorphic because they both have order 8.”
Policy proposed: group isomorphism
Status: uncertified
Immediate diagnostic: order is insufficient invariant
Next action: compare element-order spectrum or group table
Possible result: inadmissible or undecided

The proposal is useful because it triggers analysis, but it is not accepted.

Another example:

ModelOutput: “This theorem follows by compactness.”
Policy proposed: theorem application
Status: uncertified
Diagnostic: compactness hypothesis missing from Γ
Possible result: obstruction

The uncertified proposal record allows the AI to learn from its own near-misses. It preserves the gap between guess and proof.

B.6 JSON-Like Conceptual Form

A machine-readable record can be represented conceptually as follows:

{
"judgment_id": "J-000001",
"context": {
"context_id": "G-finite-groups",
"domain": "finite group theory",
"assumptions": ["groups finite"],
"formal_system": null
},
"object_x": {
"object_id": "Obj-Z4",
"name": "Z/4Z",
"kind": "group"
},
"object_y": {
"object_id": "Obj-V4",
"name": "Z/2Z × Z/2Z",
"kind": "group"
},
"policy": {
"policy_id": "Pol-GroupIso",
"name": "group isomorphism"
},
"result_status": "inadmissible",
"certificate": null,
"obstruction": {
"obstruction_id": "Obs-ElementOrder",
"kind": "invariant_mismatch",

"violated_invariant": "element-order spectrum",
"content": "Z/4Z has an element of order 4; Z/2Z × Z/2Z does not."
},
"substitution_scope": null,
"verification_status": "verified_human",
"source": "standard finite group theory",
"notes": "Same under cardinality policy, not under group-isomorphism policy."
}

A positive companion record would be:

{
"judgment_id": "J-000002",
"context": {
"context_id": "FiniteSets",
"domain": "finite set comparison"
},
"object_x": {
"object_id": "Obj-Z4",
"name": "underlying set of Z/4Z"
},
"object_y": {
"object_id": "Obj-V4",
"name": "underlying set of Z/2Z × Z/2Z"
},
"policy": {
"policy_id": "Pol-Cardinality",
"name": "cardinality equivalence"
},
"result_status": "admissible",
"certificate": {
"certificate_id": "Cert-BothFour",
"kind": "cardinality_computation",
"content": "Both underlying sets have four elements."
},
"invariants": ["cardinality"],
"substitution_scope": "finite-set cardinality arguments",
"verification_status": "verified_computational",

"notes": "Does not license group-theoretic substitution."
}

These examples show why the record must support multiple policies for the same pair. The same objects are positively related under one policy and negatively related under another.

B.7 Required Fields for Training

For machine learning, the record should be converted into training examples with explicit fields. A training example should include:

Input object X
Candidate object Y
Context Γ
Candidate policy π
Required invariants Iπ
Result status
Certificate C if positive
Obstruction O if negative
Substitution scope Sπ
Explanation
Verification status

A positive training example teaches:

Under π, X and Y are admissibly the same because C preserves Iπ.

A negative training example teaches:

Under π, X and Y are not admissibly the same because O shows invariant I fails.

An undecided training example teaches:

The system lacks C and lacks O; do not assert admissibility.

An uncertified training example teaches:

This is a proposal; search for C or O before use.

This training format differs sharply from standard input-output learning. The model is trained to classify policy, retrieve invariants, detect certificate type, identify obstructions, and decide substitution scope. It learns mathematical permission.

B.8 Required Fields for Inference

At inference time, a mathematical AI should construct or retrieve an admissibility record before making a strong claim. The minimum inference sequence is:

Identify X and Y.
Identify context Γ.
Infer candidate policies π.
Retrieve required invariants Iπ.
Search for existing judgments.
Search for certificates C.
Search for obstructions O.
Assign result status.
Return scoped answer.

If no policy can be inferred, the system should ask or present policy alternatives. For example:

Same under cardinality?
Same under group isomorphism?
Same under ring isomorphism?
Same under homotopy equivalence?
Same under formal definitional equality?

If the user's intent strongly implies a policy, the system may select it but should still expose it. Hidden policy selection is the root of many hallucinations.

At inference time, the system should avoid converting heuristic similarity into admissibility. A vector match may suggest Y as relevant to X, but it should produce an uncertified proposal record, not a positive judgment. Only certificate checking can promote it.

B.9 Promotion Rules

Records should move through states by promotion rules.

Uncertified proposal → admissible
only when certificate C is found and verified.

Uncertified proposal → inadmissible
only when obstruction O is found and verified.

Uncertified proposal → undecided
when search fails without obstruction.

Undecided → admissible
when certificate C is later found.

Undecided → inadmissible
when obstruction O is later found.

Admissible → obsolete
when context, formal system, source, or dependency changes such that the certificate must be rechecked.

Inadmissible → superseded
when the policy or context is refined and the original obstruction no longer applies to the revised claim.

These promotion rules protect the database from false certainty. They also make the research process auditable. A generated claim can mature into a verified theorem, fail by obstruction, or remain open.

B.10 Substitution Authorization

The most important operational use of the record is substitution authorization. Before using Y in place of X, the AI should check:

Is there an admissible judgment $\Gamma \vdash X \equiv_\pi Y : C$?
Is the current context compatible with Γ?
Is the current use inside substitution scope S_π?
Are the required invariants preserved?
Is the certificate still valid under current versions?
Is there any obstruction under a stronger required policy?

If all checks pass, substitution is authorized. If not, the system should refuse, qualify, or seek a stronger certificate.

This makes the admissibility record a gatekeeper. It prevents a model from using similarity as replacement. It also prevents valid equivalences from being used outside their jurisdiction.

For example, if the system wants to transport a metric property across a homeomorphism, it should check whether the property is preserved by homeomorphism. If not, the substitution fails unless a stronger policy, such as isometry, is certified.

B.11 Explanation Generation from Records

Human-readable explanations should be generated from admissibility records, not improvised independently. A record can produce a concise explanation:

“X and Y are admissibly the same under π because certificate C preserves invariants Iπ. This licenses substitution only for Sπ.”

Or:

“X and Y are not admissibly the same under π because obstruction O violates invariant I. They may still be comparable under weaker policy π′.”

This improves trust because the explanation remains anchored to structured data. The AI is not merely saying what sounds right. It is verbalizing the record.

For example:

“Z/4Z and Z/2Z × Z/2Z have the same cardinality, so they are identical under cardinality policy. They are not identical under group-isomorphism policy because element order is preserved by group isomorphism, and only Z/4Z has an element of order 4.”

This explanation is compact because the record contains the policy distinction.

B.12 The Record as Mathematical Memory

The admissibility record is mathematical memory in its proper form. It remembers not only claims, but the authority structure behind claims. It remembers the context, policy, certificate, obstruction, and scope. This is what ordinary AI memory lacks.

A mathematical memory that stores only “X and Y are equivalent” is unsafe. It may later use the equivalence under the wrong policy. A mathematical memory that stores “X and Y are equivalent under π with certificate C and scope Sπ” is usable. It tells the system what may be done.

The record also prevents identity drift. If a theorem is revised, a proof changes, a formal library updates, or a policy is refined, the database can update the record’s status. It can say that the old judgment is obsolete, superseded, or still valid. This makes long-term mathematical AI possible.

B.13 Minimal Demonstration

The simplest demonstration of the central record should involve a pair of objects with different judgments under different policies.

Objects:

$X = Z/4Z$
$Y = Z/2Z \times Z/2Z$

Policy 1:

π = cardinality equivalence
Result: admissible
Certificate: both have four elements
Scope: finite-set cardinality comparison

Policy 2:

π = group isomorphism
Result: inadmissible
Obstruction: element-order mismatch
Scope: group-theoretic substitution blocked

The AI should output:

Under cardinality policy, X and Y are admissibly the same. Under group-isomorphism policy, they are not admissibly the same. The obstruction is that group isomorphisms preserve element orders, but X has an element of order 4 while Y does not. Therefore Y may not replace X in group-theoretic arguments, though it may replace X in arguments concerning cardinality alone.

This demonstration is small but complete. It shows context, policy, invariant, certificate, obstruction, and substitution scope. It is the seed of the entire architecture.

B.14 Summary of Appendix B

The central machine-readable record is the atomic unit of Admissibility Algebra for Mathematical AI. It has four principal states: certified admissibility, certified inadmissibility, undecided, and uncertified proposal. It records context, objects, policy, invariants, certificate, obstruction, substitution scope, verification status, and provenance. It allows mathematical AI to distinguish proof from plausibility, obstruction from failed search, and lawful substitution from superficial resemblance.

The record converts the paper's thesis into a computational object. Future AI should not merely store facts or generate answers. It should store admissibility judgments: what may count as the same, under which policy, preserving what, with what witness, and where the substitution is allowed.

Appendix C. Minimal Working Prototype

C.1 Purpose of the Prototype

The minimal working prototype is not intended to solve theorem proving, replace proof assistants, train a large model, or cover all of mathematics. Its purpose is narrower and more important: to demonstrate that policy-indexed identity can be represented, queried, and explained by a machine. The prototype should prove that the central judgment of Admissibility Algebra can become a working computational object.

The prototype should answer questions of the following form:

Are X and Y the same?

But instead of returning a single yes or no, it should return a policy-indexed identity profile:

Same under π_1 with certificate C_1.
Not same under π_2 because obstruction O_2.
Undecided under π_3.
Uncertified under π_4.
Substitution licensed only in scope $S\pi$.

This is the essential behavior. A conventional AI system may answer from memory or by fluent explanation. The prototype must answer from structured admissibility records. It should expose the policy, invariant, certificate, obstruction, and scope.

The first prototype should therefore be small, deterministic, inspectable, and mathematically obvious. Its purpose is to establish the architecture, not impress with scale. A tiny corpus of carefully chosen examples is preferable to a large unverified corpus. The system should show that it can avoid the central error of current AI: treating resemblance as identity.

The ideal first demonstration is the comparison between $\mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. The pair is simple, familiar, and decisive. They are the same under cardinality policy because both have four elements. They are not the same under group-isomorphism policy because element-order structure differs. This single example demonstrates multi-policy identity, positive certification, negative obstruction, invariant preservation, and scoped substitution.

A second demonstration may compare two finite-dimensional vector spaces over the same field, emphasizing noncanonical isomorphism. A third may compare two spaces that are homotopy equivalent but not homeomorphic. A fourth may compare two theorem statements that are semantically similar but not logically equivalent. But the first demonstration should remain minimal.

The prototype is successful if it can do five things. First, store objects without collapsing them. Second, store policies as first-class entities. Third, attach invariants to policies. Fourth, return certificates and obstructions. Fifth, explain the substitution scope. These five capabilities are the seed of mathematical admissibility AI.

C.2 Prototype Architecture

The minimal prototype can be implemented as a small local system with four layers:

Structured store
Policy graph
Retrieval interface
Explanation generator

The structured store holds objects, policies, invariants, certificates, obstructions, and judgments. It may begin as a simple SQLite or PostgreSQL database. The first version does not require a large graph database or vector index. It needs correctness more than scale. Each judgment should be explicitly entered and queryable.

The policy graph records relationships among policies. For the minimal prototype, this graph may be small. It can include cardinality equivalence, set bijection, group isomorphism, ring isomorphism, homeomorphism, homotopy equivalence, definitional equality, propositional equality, and heuristic similarity. The graph should record refinement and weakening relationships where obvious. For example, group isomorphism implies cardinality equivalence for finite groups, but cardinality equivalence does not imply group isomorphism.

The retrieval interface receives a user query and identifies candidate objects and policies. In the first version, this can be rule-based. The user may ask explicitly: “Are $\mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ the same as groups?” The system should map “as groups” to group-isomorphism policy. If the user asks simply “are they the same?” the system should return multiple policies rather than guessing one.

The explanation generator converts admissibility records into human-readable prose. It should not invent explanations independently. It should verbalize the structured record. For a positive judgment, it should state the policy, certificate, preserved invariant, and substitution scope. For a negative judgment, it should state the policy, obstruction, violated invariant, and possible weaker policies. For undecided records, it should state what is missing.

The architecture can be represented as:

Input query
Object identification
Policy selection or multi-policy expansion

Judgment lookup
Certificate or obstruction retrieval
Substitution-scope check
Human-readable explanation

The first prototype does not need machine learning. It may be entirely symbolic and database-driven. This is important. The goal is to build the admissibility skeleton before adding neural proposal. A neural model can be attached later to suggest objects, policies, and candidate certificates, but the minimal system should first prove that admissibility records are useful.

The prototype should also separate verified records from generated suggestions. If a language model is later used to propose a comparison, the proposal should enter the database as uncertified. It should not be promoted to admissible or inadmissible without a certificate or obstruction.

The guiding design principle is:

No hidden sameness.

Every answer must expose the policy under which sameness is being asserted.

C.3 Minimal Corpus

The first corpus should be deliberately small. It should contain enough examples to show that the same objects can have different identity statuses under different policies. It should also include examples of positive certificates, negative obstructions, undecided cases, and uncertified proposals.

A minimal corpus may include the following object pairs:

$\mathbb{Z}/4\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$
Finite sets with four elements
Two cyclic groups of order 4
Two vector spaces of dimension n over the same field
A vector space V and k^n after choosing a basis
Two categories that are equivalent but not strictly identical
Two spaces with the same homology but not known equivalent
Two expressions equal on a restricted domain but not globally
Two theorem statements differing by a missing hypothesis
A quotient by a normal subgroup
A proposed quotient by a non-normal subgroup

For each pair, the corpus should contain multiple policies. The purpose is to teach the system that identity is indexed. A pair should not be labeled simply same or different. It should have an identity profile.

Example identity profile:

Objects: $Z/4Z$, $Z/2Z \times Z/2Z$

Policy: cardinality equivalence
Result: admissible
Certificate: both have four elements
Substitution scope: finite-set cardinality arguments

Policy: group isomorphism
Result: inadmissible
Obstruction: element-order invariant mismatch
Substitution scope: none for group-theoretic replacement

Policy: abelian group comparison
Result: both abelian
Certificate: operation commutative in both
Warning: shared abelian property does not certify group isomorphism

This last record is important because it distinguishes shared invariant from complete certificate. Both groups are abelian, but abelianity alone does not make them isomorphic. The prototype should be able to say that a shared invariant is evidence or common property, not full admissibility.

Another profile:

Objects: V, k^n

Policy: vector-space isomorphism
Result: admissible if $\dim V = n$ and a basis is chosen or existence is accepted
Certificate: basis-dependent linear equivalence
Canonicality: noncanonical unless basis specified
Substitution scope: vector-space properties

Policy: equality of presentations
Result: inadmissible or undecided unless V is definitionally represented as k^n
Obstruction: different presentation or missing definitional equality

This teaches the system about noncanonical certificates.

Another profile:

Objects: informal theorem T, formal theorem F

Policy: formalization fidelity
Result: undecided unless annotated
Possible certificate: proof alignment and hypothesis preservation
Possible obstruction: missing hypothesis, changed quantifier, changed domain
Substitution scope: citing F as formal version of T only after fidelity certificate

This teaches that formal proof of F does not automatically prove that F faithfully represents T.

The minimal corpus should be hand-curated. Automated ingestion can come later. At the beginning, quality matters more than quantity.

C.4 Minimal Policies

The first prototype needs a small set of policies. Each policy should be defined with certificate type, invariant set, obstruction type, and substitution scope.

Policy 1: literal equality
Certificate: equality proof or identical formal object
Invariants: all structure visible in context
Obstruction: syntactic or formal inequality, type mismatch
Scope: unrestricted only within the formal context

Policy 2: definitional equality
Certificate: reduction, normalization, kernel recognition
Invariants: computational meaning under formal system
Obstruction: failure to reduce to same normal form
Scope: proof-assistant computation and rewriting

Policy 3: cardinality equivalence
Certificate: bijection or equal finite count
Invariants: number of elements
Obstruction: different cardinality
Scope: finite-set size arguments only

Policy 4: group isomorphism
Certificate: bijective group homomorphism or group equivalence data
Invariants: group operation, identity, inverses, element orders, group-theoretic properties
Obstruction: order mismatch, element-order mismatch, commutativity mismatch, subgroup-

structure mismatch
Scope: group-theoretic substitution

Policy 5: vector-space isomorphism
Certificate: linear equivalence
Invariants: linear structure, dimension
Obstruction: dimension mismatch, field mismatch, failure of linearity
Scope: vector-space properties

Policy 6: homeomorphism
Certificate: continuous bijection with continuous inverse
Invariants: topological properties
Obstruction: compactness mismatch, connectedness mismatch, topological invariant mismatch
Scope: topological substitution

Policy 7: homotopy equivalence
Certificate: maps both ways plus homotopies to identities
Invariants: homotopy-invariant properties
Obstruction: homology mismatch, fundamental-group mismatch, homotopy-invariant mismatch
Scope: homotopy-theoretic substitution

Policy 8: categorical equivalence
Certificate: functors plus natural isomorphism data, or full faithful essentially surjective functor
Invariants: categorical structure up to equivalence
Obstruction: failure of fullness, faithfulness, essential surjectivity, naturality, coherence
Scope: equivalence-invariant categorical reasoning

Policy 9: formalization fidelity
Certificate: human or formal alignment of informal and formal statement
Invariants: hypotheses, quantifiers, definitions, domains, conclusions, intended generality
Obstruction: missing hypothesis, altered quantifier, changed domain, definitional mismatch
Scope: treating formal statement as representative of informal statement

Policy 10: heuristic similarity
Certificate: embedding proximity or model-generated resemblance
Invariants: none guaranteed
Obstruction: not applicable; heuristic only
Scope: retrieval and search, not proof substitution

The last policy is essential. It allows the system to use vector search while preventing vector search from certifying identity. Heuristic similarity is useful, but it is not admissibility unless upgraded by another policy.

The minimal prototype should make this distinction explicit in every answer. If a record comes from heuristic similarity, the system should say: related for search, not certified for substitution.

C.5 Minimal Invariants

The first invariant set should be small, concrete, and easy to explain.

Cardinality
Element-order spectrum
Commutativity
Dimension
Field of scalars
Continuity
Compactness
Connectedness
Homology group
Fundamental group
Naturality
Functoriality
Hypothesis preservation
Quantifier preservation
Domain preservation
Well-definedness
Congruence compatibility
Type correctness
Definitional reduction

Each invariant should be linked to policies that preserve it. Cardinality is preserved by bijection and by finite group isomorphism. Element-order spectrum is preserved by group isomorphism. Dimension is preserved by vector-space isomorphism. Compactness is preserved by homeomorphism but not by arbitrary homotopy equivalence. Homology is preserved by homotopy equivalence. Hypothesis preservation is required by formalization fidelity. Well-definedness is required for quotient policies. Type correctness is required by formal proof transformations.

The prototype should distinguish invariant match from certificate. If two groups both have four elements, that matches cardinality but does not certify group isomorphism. If two spaces have the same homology, that matches a homotopy invariant but does not necessarily certify homotopy equivalence. This distinction is central to preventing AI overclaiming.

A useful invariant record should include:

Invariant name
Definition
Policies preserving it
Policies for which mismatch obstructs
Whether matching is complete for any class
Computation method
Example positive use
Example obstruction

For example:

Invariant: element-order spectrum
Definition: multiset or set of orders of elements in a group
Preserved by: group isomorphism
Mismatch obstructs: group isomorphism
Complete for: not complete in general
Example obstruction: Z/4Z vs Z/2Z × Z/2Z

This form teaches the AI that mismatch is decisive, but match may not be complete.

C.6 Minimal Queries

The prototype should support a few basic query types.

Query type 1: Direct comparison

Input: Are X and Y the same?
Output: multi-policy identity profile.

Example output:

Under cardinality policy, yes. Under group-isomorphism policy, no. The obstruction is element-order mismatch. Therefore substitution is allowed only for cardinality arguments, not group-theoretic arguments.

Query type 2: Policy-specific comparison

Input: Are X and Y isomorphic as groups?
Output: admissible, inadmissible, undecided, or uncertified under group-isomorphism policy.

Query type 3: Weakest policy search

Input: Under what policy are X and Y the same?
Output: list of policies under which admissibility is certified, plus stronger policies that fail or remain undecided.

Query type 4: Obstruction search

Input: Why are X and Y not the same under π?
Output: violated invariant and obstruction proof.

Query type 5: Certificate search

Input: What certifies that X and Y are the same under π?
Output: certificate type and content.

Query type 6: Substitution check

Input: May I replace X by Y in context K?
Output: yes only if K lies inside substitution scope Sπ; otherwise no or requires stronger policy.

Query type 7: Quotient validity

Input: Can I form Ω/π?
Output: yes if relation is compatible and well-defined; no if congruence or well-definedness fails; undecided if no certificate.

These queries are enough to show that the prototype is not a chatbot but an admissibility engine. It is answering permission questions.

C.7 Example Dialogue

A minimal demonstration dialogue should look like this.

User:

Are Z/4Z and Z/2Z × Z/2Z the same?

System:

They are the same under cardinality policy because both underlying sets have four elements. They are not the same under group-isomorphism policy. The obstruction is element-order structure: Z/4Z contains an element of order 4, while every nonidentity element of Z/2Z × Z/2Z has order 2. Therefore one may substitute them in finite-set cardinality arguments, but not in group-theoretic arguments.

User:

So they are equivalent?

System:

Equivalent under which policy? Under cardinality equivalence, yes. Under group isomorphism, no. Under abelian-group property comparison, both are abelian, but that shared invariant does not certify isomorphism.

User:

What is the weakest policy under which they are the same?

System:

The weakest recorded positive policy is cardinality equivalence. They share other coarse properties, such as being finite abelian groups of order 4, but they are not isomorphic as groups. A classification policy for finite abelian groups distinguishes them by invariant factor decomposition.

User:

Can I replace one by the other in a proof about finite abelian groups?

System:

Only if the proof uses properties preserved by the policy you choose. If the proof uses only cardinality or abelianity, replacement may be allowed under that limited scope. If the proof uses group structure up to isomorphism, replacement is not allowed.

This dialogue demonstrates the entire point. The system does not merely answer. It governs substitution.

C.8 Minimal Implementation Without Programming Burden

The first prototype can be built by a programmer or collaborator, but the mathematical author does not need to become the programmer. The specification can be handed off as a small implementation task.

A minimal implementation can use:

SQLite or PostgreSQL for structured records
A small Python or web interface
Hand-entered JSON/YAML records for objects and policies
No large language model initially
Optional local LLM only for natural-language paraphrase
A simple rule engine for policy lookup
Manual verification status labels

The author's role should be to supply examples, definitions, policies, invariants, certificates, and obstructions. The implementer's role should be to encode them. This preserves conceptual productivity while allowing the prototype to exist.

The first deliverable from an implementer should not be a large system. It should be a working demo with perhaps twenty objects, ten policies, thirty invariants, and fifty judgments. If the demo can answer policy-indexed sameness questions correctly, it proves the architecture.

The prototype should output structured JSON and human-readable prose. The JSON demonstrates machine-readability. The prose demonstrates usability.

C.9 Success Criteria

The prototype succeeds if it satisfies the following criteria.

It never answers "same" without naming a policy.
It can give different judgments for the same pair under different policies.
It distinguishes certificate from explanation.
It distinguishes obstruction from failed search.
It distinguishes heuristic similarity from admissibility.
It states substitution scope.
It can return the weakest recorded policy under which identity holds.
It can say undecided without pretending to know.
It can store and retrieve quotient records.
It can explain why a substitution is unauthorized.

The Z/4Z example is the basic test. If the system says simply "no, they are different," it fails because it missed cardinality policy. If it says simply "yes, they both have four elements," it fails because it missed group-isomorphism obstruction. If it says "they are related," it fails because it did not state policy. The correct answer is multi-policy.

A stronger test is formalization fidelity. Give the system two theorem statements, one missing a hypothesis. It should not say they are equivalent because they sound similar. It should identify the missing hypothesis as an obstruction under semantic or formalization-fidelity policy.

A third test is quotient formation. Give the system a proposed group quotient by a non-normal subgroup. It should identify non-normality as an obstruction to quotient group admissibility.

These tests are small but decisive. They show whether the system has learned permission rather than resemblance.

C.10 Path to the Next Prototype

After the minimal prototype works, the next version can add:

Lean/mathlib extraction
Formal proof checking
Vector retrieval for candidate theorem search
Graph database for policy lattice
Tactic trace ingestion
Automated invariant computation for small algebraic examples
Model-generated uncertified proposal records
Human review interface
Exportable admissibility training data

The order matters. The database and policy logic should precede large-scale AI. Otherwise the system will merely generate more unstructured mathematical text. The admissibility corpus is the asset. Machine learning becomes useful after the corpus exists.

The second prototype should connect to a proof assistant. It should take a proposed formal theorem or rewrite and ask Lean or another system to check it. Successful checks become certificates. Failed checks become proof traces or obstructions if diagnosed. This adds formal authority.

The third prototype should add retrieval. It may use embeddings to suggest candidate theorems, but every retrieved result should enter as heuristic until certified. This preserves the distinction between search and proof.

The fourth prototype should train a small model on admissibility records. The goal is not to produce a general theorem prover. The goal is to learn policy classification, invariant retrieval, obstruction diagnosis, and certificate-type prediction.

C.11 Summary of Appendix C

The minimal working prototype is a small admissibility engine. It stores objects, policies, invariants, certificates, obstructions, quotients, and judgments. It answers sameness questions by returning policy-indexed identity profiles. It distinguishes positive certificates, negative obstructions, undecided cases, and uncertified proposals. It exposes substitution scope. It treats heuristic similarity as search, not proof.

The prototype should begin with simple examples, especially $\mathbb{Z}/4\mathbb{Z}$ versus $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. This example demonstrates the entire architecture: same under cardinality, not same under group isomorphism, obstruction by element-order invariant, substitution only within scope.

The purpose of the prototype is not scale. It is discipline. It proves that future mathematical AI can be built around admissible transformation rather than uncontrolled generation.

Appendix D. First Demonstration: Z/4Z and Z/2Z × Z/2Z

D.1 Why This Example Matters

The comparison between Z/4Z and Z/2Z × Z/2Z is the smallest useful demonstration of Admissibility Algebra. It is elementary enough to be understood without advanced machinery, yet rich enough to expose the central error that mathematical AI must avoid: treating one kind of sameness as if it were another.

Both objects have four elements. Both are finite groups. Both are abelian. Both may appear in the same chapter of an abstract algebra text. Both may be described as groups of order 4. Under several weak policies, they are related. But they are not isomorphic as groups. The difference is decisive: Z/4Z has an element of order 4, while Z/2Z × Z/2Z does not. Every nonidentity element of Z/2Z × Z/2Z has order 2.

This example therefore forces the system to separate policies. Under cardinality policy, the two objects are admissibly the same. Under finite-set bijection, they are admissibly the same. Under abelianity-as-shared-property, they share an invariant but are not thereby identified. Under group-isomorphism policy, they are inadmissible as replacements for one another. The obstruction is element-order structure.

A conventional AI might answer in one of several incomplete ways. It might say “they are different,” missing the weaker policies under which they are the same. It might say “they are both groups of order 4,” implying a stronger sameness than is warranted. It might say “they are not isomorphic” without explaining the substitution scope. It might say “they are equivalent as finite abelian groups” ambiguously, confusing shared classification context with actual group isomorphism. An admissibility-aware system should avoid all of these.

The correct answer is policy-indexed:

They are the same under cardinality policy.
They are not the same under group-isomorphism policy.
The obstruction is element-order mismatch.
Substitution is licensed only in contexts preserving the selected policy.

This example is minimal, but it contains the whole architecture: objects, policies, invariants, certificates, obstructions, judgments, quotient-like forgetting, and scoped substitution.

D.2 The Objects

Let X = Z/4Z. This is the additive cyclic group of integers modulo 4. Its elements are:

0, 1, 2, 3

with addition performed modulo 4. The element 1 generates the group:

1 + 1 = 2
1 + 1 + 1 = 3
1 + 1 + 1 + 1 = 0

Thus 1 has order 4. The element 3 also has order 4. The element 2 has order 2. The identity 0 has order 1. The element-order profile is therefore:

one element of order 1
one element of order 2
two elements of order 4

Let Y = Z/2Z × Z/2Z. This is the direct product of two cyclic groups of order 2. Its elements are:

(0,0), (1,0), (0,1), (1,1)

with coordinate-wise addition modulo 2. The identity is (0,0). Every nonidentity element has order 2:

(1,0) + (1,0) = (0,0)
(0,1) + (0,1) = (0,0)
(1,1) + (1,1) = (0,0)

The element-order profile is therefore:

one element of order 1
three elements of order 2
zero elements of order 4

The two objects have the same number of elements, but different group structure. The distinction is invisible under cardinality and visible under group isomorphism. This is exactly why the example is useful.

The prototype should store X and Y as separate MathematicalObject records. It should not merge them merely because they have the same size or because both are abelian groups of order 4. Their relation must be represented through policy-indexed judgments.

D.3 Policy 1: Cardinality Equivalence

Under cardinality equivalence, X and Y are admissibly the same. The policy πcard asks only whether the underlying finite sets have the same number of elements. It does not ask whether operations, generators, element orders, subgroup structure, or algebraic presentation are preserved.

The admissibility judgment is:

$\Gamma \vdash X \equiv_{\pi\text{card}} Y : C_{\text{card}}$

where Γ is the context of finite-set comparison, π_{card} is cardinality equivalence, and C_{card} is the certificate that both underlying sets have four elements.

The invariant preserved by π_{card} is cardinality. The certificate is simple:

$|X| = 4$
$|Y| = 4$
therefore $|X| = |Y|$

If desired, a stronger set-level certificate may be supplied by an explicit bijection:

$0 \mapsto (0,0)$
$1 \mapsto (1,0)$
$2 \mapsto (0,1)$
$3 \mapsto (1,1)$

This bijection certifies equality of cardinality or finite-set equivalence. But it is not a group isomorphism. It does not preserve the group operation. For example, in $Z/4Z$:

$1 + 1 = 2$

Under the displayed bijection:

$1 \mapsto (1,0)$
$1 \mapsto (1,0)$
$2 \mapsto (0,1)$

But:

$(1,0) + (1,0) = (0,0)$

which is not $(0,1)$. Therefore this bijection is not structure-preserving as a group map.

The substitution scope $S_{\pi\text{card}}$ is narrow. Under cardinality policy, Y may replace X only in arguments that depend solely on the number of elements. For example, statements such as “there exists a finite set with four elements” or “the underlying set has cardinality 4” permit replacement. Group-theoretic claims do not.

The database record should therefore state:

Result: admissible under cardinality policy.
Certificate: both have four elements.

Invariant: cardinality.
Substitution scope: finite-set size arguments only.
Warning: does not license group-theoretic substitution.

This warning is not optional. It is the difference between correct abstraction and destructive collapse.

D.4 Policy 2: Group-Isomorphism

Under group-isomorphism policy, X and Y are not admissibly the same. The policy πgrp asks whether there exists a bijective homomorphism preserving the group operation. For additive groups, a map f must satisfy:

$f(a + b) = f(a) + f(b)$

and must be bijective.

The admissibility failure is:

$\Gamma \nvdash X \equiv_{\pi grp} Y$ because $O_{\pi grp}(X,Y)$

where Oπgrp is the element-order obstruction.

A group isomorphism preserves element order. If $f: G \to H$ is a group isomorphism and $g \in G$ has order n, then f(g) has order n. This is because:

$f(g^n) = f(e) = e$

and if $f(g)^k = e$, then $f(g^k) = e$, hence $g^k = e$ by injectivity. Therefore the least positive exponent returning g to the identity is preserved.

In $X = Z/4Z$, there is an element of order 4. The element 1 has order 4. In $Y = Z/2Z \times Z/2Z$, no element has order 4. Every nonidentity element has order 2. Therefore no group isomorphism can exist.

The obstruction record is:

Invariant: element-order spectrum
Preserved by: group isomorphism
Mismatch: X has elements of order 4; Y has none
Conclusion: inadmissible under group-isomorphism policy

This is a certified negative result. It is not a failure to find an isomorphism. It is a proof that no such isomorphism can exist.

The substitution scope under πgrp is therefore empty for replacing X by Y as groups. One cannot use Y in place of X in a group-theoretic argument requiring isomorphism. Statements about cyclicity, generators, element orders, subgroup behavior, and classification of finite abelian groups distinguish them.

For example, X is cyclic. Y is not cyclic. X has a generator. Y has no element generating all four elements. Therefore any proof depending on cyclicity cannot replace X by Y.

The database should answer:

Not admissible under group-isomorphism policy. The obstruction is element-order mismatch. Since element order is preserved by group isomorphism, and only Z/4Z contains an element of order 4, no group isomorphism exists.

This is the first negative test for the prototype.

D.5 Policy 3: Shared Abelian Property

Both X and Y are abelian. This is a true shared property, but it is not an identity certificate. The prototype should represent this carefully because it is a common source of AI overclaiming.

The relevant statement is:

X satisfies invariant A.
Y satisfies invariant A.
A = abelian group property.

This does not imply:

X ≡πgrp Y.

Shared invariants may support comparison, classification, or search, but they do not necessarily certify admissibility. Many nonisomorphic groups are abelian. Many nonisomorphic groups have the same order. Many nonhomeomorphic spaces have the same homology. Many non-equivalent theorem statements share terminology. The AI must learn that shared features are not automatically certificates.

The shared-abelian record should therefore be stored as an invariant comparison, not as an admissibility judgment under group isomorphism. It may support a weaker policy such as “both belong to the class of finite abelian groups of order 4,” but this class-membership policy is not replacement identity.

The system should say:

Both objects are abelian groups of order 4, but this shared classification does not certify group isomorphism. The group-isomorphism policy is blocked by element-order mismatch.

This is a crucial behavior. It prevents the model from turning a common property into a false equivalence.

D.6 Policy 4: Classification of Finite Abelian Groups

The classification theorem for finite abelian groups gives a richer context. Groups of order 4 have two abelian isomorphism types:

$\mathbb{Z}/4\mathbb{Z}$
$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$

This classification does not identify the two groups. It distinguishes them. It places them in the same classification problem while assigning them to different isomorphism classes.

This is a useful example of quotient reasoning. Let Ω be the class of abelian groups of order 4. Let π_{iso} be group isomorphism. Then Ω/π_{iso} has two equivalence classes:

$[\mathbb{Z}/4\mathbb{Z}]$
$[\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}]$

Thus X and Y are elements of the same ambient domain Ω, but they occupy different quotient classes under π_{iso}.

The prototype should be able to express:

X and Y both lie in Ω = finite abelian groups of order 4.
Under the quotient Ω/π_{iso}, they are distinct equivalence classes.
The invariant distinguishing them is cyclic decomposition or element-order structure.

This prevents another common confusion: belonging to the same classification theorem does not mean belonging to the same equivalence class. A theorem may classify multiple objects precisely by distinguishing them.

The classification context also illustrates weakest-policy reasoning. If the question is “are they both abelian groups of order 4?” the answer is yes. If the question is “are they the same isomorphism type?” the answer is no. If the question is “do they appear in the same finite abelian classification problem?” the answer is yes. Each question selects a different policy.

D.7 Machine-Readable Records

The first prototype should store at least four records for this example.

Record 1: Object X

ObjectID: Obj-Z4
Name: Z/4Z
Kind: finite abelian group
Presentation: cyclic group of order 4
Elements: 0,1,2,3
Operation: addition modulo 4
Properties: order 4, cyclic, abelian, contains element of order 4

Record 2: Object Y

ObjectID: Obj-V4
Name: Z/2Z × Z/2Z
Kind: finite abelian group
Presentation: direct product of two cyclic groups of order 2
Elements: (0,0),(1,0),(0,1),(1,1)
Operation: coordinate-wise addition modulo 2
Properties: order 4, noncyclic, abelian, all nonidentity elements order 2

Record 3: Positive cardinality judgment

JudgmentID: J-Z4-V4-cardinality
Context: finite-set comparison
ObjectX: Obj-Z4 underlying set
ObjectY: Obj-V4 underlying set
Policy: cardinality equivalence
ResultStatus: admissible
Certificate: both have four elements
Invariant: cardinality
SubstitutionScope: finite-set cardinality arguments
VerificationStatus: verified elementary computation

Record 4: Negative group-isomorphism judgment

JudgmentID: J-Z4-V4-groupiso
Context: finite group theory
ObjectX: Obj-Z4
ObjectY: Obj-V4

Policy: group isomorphism
ResultStatus: inadmissible
Obstruction: element-order mismatch
ViolatedInvariant: element-order spectrum
Evidence: Obj-Z4 has elements of order 4; Obj-V4 has none
SubstitutionScope: no group-theoretic replacement
VerificationStatus: verified elementary proof

Record 5: Shared property record

ComparisonID: C-Z4-V4-abelian
Context: finite group theory
ObjectX: Obj-Z4
ObjectY: Obj-V4
SharedInvariant: abelianity
Result: both satisfy property
Warning: shared invariant does not certify group isomorphism

Record 6: Classification quotient record

QuotientID: Q-AbelianGroupsOrder4-Iso
BaseDomain: abelian groups of order 4
Policy: group isomorphism
QuotientClasses: [Z/4Z], [Z/2Z × Z/2Z]
DistinguishingInvariant: cyclic decomposition or element-order structure
Status: certified by finite abelian group classification

These records are enough to generate correct answers.

D.8 Expected Prototype Responses

The prototype should answer several queries in a policy-indexed way.

Query:

Are Z/4Z and Z/2Z × Z/2Z the same?

Expected response:

They are the same under cardinality policy because both underlying sets have four elements. They are not the same under group-isomorphism policy because element-order structure differs: Z/4Z has an element of order 4, while Z/2Z × Z/2Z has no element of order 4. Therefore one may replace one with the other only in cardinality arguments, not in group-theoretic arguments.

Query:

Are they isomorphic?

Expected response:

No, not as groups. A group isomorphism preserves element orders. Z/4Z contains elements of order 4, but every nonidentity element of Z/2Z × Z/2Z has order 2. This obstructs group isomorphism.

Query:

But they both have four elements, right?

Expected response:

Yes. They are admissibly the same under cardinality equivalence. That policy preserves only the number of elements. It does not preserve group operation or element-order structure, so it does not license group-theoretic substitution.

Query:

Can I use Z/2Z × Z/2Z instead of Z/4Z in a proof about finite abelian groups?

Expected response:

Only if the proof uses properties shared under the selected policy. If the proof uses only cardinality or abelianity, the substitution may be allowed in that limited scope. If the proof uses cyclicity, generators, element orders, or group-isomorphism type, the substitution is not allowed.

Query:

What is the obstruction?

Expected response:

The obstruction is the element-order invariant. Element orders are preserved under group isomorphism. Z/4Z has an element of order 4; Z/2Z × Z/2Z does not. Therefore no group isomorphism exists.

Query:

What quotient world separates them?

Expected response:

In the quotient of abelian groups of order 4 by group isomorphism, Z/4Z and Z/2Z × Z/2Z lie in distinct equivalence classes. The quotient has two classes, corresponding to the cyclic and noncyclic abelian groups of order 4.

These responses are short, but they demonstrate a deep architecture. The system is not merely recalling algebra. It is enforcing admissibility.

D.9 Why This Example Detects AI Failure

This example is an excellent diagnostic for mathematical AI because it exposes several common failure modes.

First, the model may overgeneralize from cardinality. It sees both objects have four elements and treats them as interchangeable. This is a policy-strength error.

Second, the model may overgeneralize from abelianity. It sees both are abelian groups and implies they are equivalent as groups. This is a shared-invariant error.

Third, the model may answer “not the same” without policy qualification. This is also incomplete because they are the same under weaker policies. It fails to represent multi-policy identity.

Fourth, the model may confuse isomorphism with classification membership. It may say they are “both finite abelian groups of order 4” as if that were a sameness certificate. This is a class-membership versus identity error.

Fifth, the model may omit substitution scope. It may correctly say they are not isomorphic but fail to state what kinds of replacement remain valid. This makes the answer less useful for proof.

Sixth, the model may give a correct conclusion but no obstruction. It may say “not isomorphic” without explaining element-order preservation. Such an answer may be factually right but not certificate-grounded.

The admissibility prototype should pass all six tests. It should not merely know the algebra result. It should know why the result depends on policy.

D.10 Extension to Other Minimal Examples

Once the Z/4Z example works, the same pattern can be extended.

Example 1: finite sets A and B with four elements.
Same under cardinality and bijection. No extra structure.

Example 2: two vector spaces V and W over the same field.
Same under vector-space isomorphism if dimensions match. Certificate may require basis or linear equivalence. Not necessarily same as inner-product spaces unless inner product preserved.

Example 3: R^2 as vector space and R^2 as Euclidean inner-product space.
Vector-space isomorphism may hold while isometry requires preservation of inner product.

Example 4: a circle and an ellipse.
Homeomorphic as topological spaces. Not equal as subsets. Not necessarily isometric under inherited metric. Same under one policy, different under another.

Example 5: a circle and a point.
Not homeomorphic. Not homotopy equivalent. Homology distinguishes them.

Example 6: a disk and a point.
Homotopy equivalent but not homeomorphic. Substitution valid for homotopy invariants, not point-set topology.

Example 7: theorem with compactness hypothesis and theorem without it.
Similar text, different admissibility under theorem-equivalence policy. Missing compactness obstructs application.

Example 8: quotient group by normal subgroup versus quotient by non-normal subgroup.
Well-defined quotient in first case; obstruction in second.

Each example teaches the same lesson: sameness is indexed by policy.

D.11 Demonstration Script

A short public demonstration can be staged as follows.

First, show the two objects:

$Z/4Z = \{0,1,2,3\}$ under addition mod 4.
$Z/2Z \times Z/2Z = \{(0,0),(1,0),(0,1),(1,1)\}$ under coordinate-wise addition mod 2.

Second, ask the audience:

Are these the same?

Third, show three answers:

As finite sets: yes, both have four elements.
As abelian groups: both belong to the class of abelian groups of order 4.
As groups up to isomorphism: no, element-order structure differs.

Fourth, state the admissibility judgment:

Γ ⊢ X ≡πcard Y : Ccard
Γ ⊬ X ≡πgrp Y because Oorder(X,Y)

Fifth, explain substitution:

You may replace one by the other in a statement about having four elements. You may not replace one by the other in a statement about cyclic groups, generators, or group isomorphism.

Sixth, give the AI lesson:

A model that says only “same” or only “different” has failed to index identity by policy. A trustworthy mathematical AI must return the policy profile.

This demonstration is suitable for a paper, lecture, or software demo because it is simple but complete.

D.12 Connection to the Main Thesis

The Z/4Z example is not a side illustration. It is the smallest working model of the entire paper.

Algebra supplies objects and operations.
Abstract algebra supplies structure-preserving maps.
Category theory generalizes morphisms and equivalences.
HoTT internalizes identity and transport.
Admissibility Algebra asks which identity policy licenses which substitution.

In this example, cardinality policy licenses one substitution scope. Group-isomorphism policy blocks another. The obstruction is invariant loss. The quotient Ω/πgrp separates the two isomorphism classes. The admissibility database stores both positive and negative judgments. The mathematical AI learns not merely to answer, but to govern replacement.

This is the essential movement from fluent generation to proof-grounded transformation. The system does not ask only whether two expressions are related. It asks what relation matters, what must be preserved, what certificate exists, what obstruction blocks stronger identity, and where substitution is allowed.

D.13 Summary of Appendix D

The first demonstration compares Z/4Z with Z/2Z × Z/2Z. They are admissibly the same under cardinality policy because both have four elements. They are inadmissible under group-isomorphism policy because element-order structure differs. Z/4Z has elements of order 4; Z/2Z × Z/2Z has none.

This example proves the need for policy-indexed identity. A single yes-or-no answer is mathematically incomplete. The correct answer is a structured admissibility profile containing policy, invariant, certificate, obstruction, and substitution scope.

If the prototype can represent and explain this example correctly, it has implemented the core of Admissibility Algebra.

Appendix E. Public Slogan and Research Program

E.1 The Public Slogan

The public slogan for Admissibility Algebra for Mathematical AI is:

Current AI predicts transformations. Mathematical AI must certify admissible transformations.

This sentence compresses the thesis of the paper into a form suitable for readers outside formal mathematics. It distinguishes present AI from the desired next stage without denying the usefulness of current systems. Current AI is powerful because it predicts plausible continuations, rewrites, summaries, answers, code, proofs, analogies, and formalizations. But prediction is not certification. A predicted transformation may be useful, suggestive, or beautiful while still being unauthorized, false, under-specified, or outside its substitution scope.

The slogan also identifies the missing layer. The future of mathematical AI is not merely larger models, longer contexts, stronger retrieval, or more fluent theorem prose. The missing layer is admissibility. The system must know when one object may replace another, when a theorem applies, when a proof step is valid, when a quotient is lawful, when a summary preserves content, when a formalization preserves meaning, and when an analogy is only an analogy. This requires policies, invariants, certificates, obstructions, and substitution scopes.

The slogan is deliberately operational. It does not say "AI must be truthful" in a vague moral sense. It says AI must certify transformations. Every serious mathematical act is a transformation: replacing an expression, applying a theorem, passing to a quotient, changing coordinates, selecting a normal form, transporting along an equivalence, summarizing a proof, formalizing a statement, or collapsing examples into a concept. The question is whether the transformation is admissible.

The slogan also provides a clean contrast:

Prediction asks: what output is likely?
Admissibility asks: what replacement is licensed?
Prediction generates Y from X.
Admissibility proves $\Gamma \vdash X \equiv_\pi Y : C$.
Prediction may be fluent.
Admissibility must be witnessed.

This contrast can guide readers, implementers, reviewers, and future AI designers. It makes the paper's technical apparatus understandable: the policies π, invariants I_π, certificates C, obstructions O, quotients Ω/π, and scopes S_π are not decorative formalism. They are the machinery required to convert predicted transformation into certified mathematical permission.

E.2 Why the Slogan Matters

The slogan matters because it reframes the problem of hallucination. In ordinary language, hallucination sounds like invention from nowhere. But many mathematical hallucinations are not pure inventions. They are unauthorized transformations. The model recognizes a pattern, retrieves a related theorem, imitates a valid proof, or compresses a statement, but it fails to check whether the transformation preserves the required identity.

For example, a model may transform a theorem about compact spaces into a theorem about arbitrary spaces. It may transform a group homomorphism into an isomorphism. It may transform categorical equivalence into strict equality. It may transform semantic similarity into theorem applicability. It may transform a proof sketch into a proof object. These are all predicted transformations. They become hallucinations when presented as admissible without certificate.

The slogan also matters because it gives a positive research direction. The goal is not simply to make models less wrong. The goal is to build systems that manage mathematical permission. A model may continue to generate, propose, conjecture, retrieve, summarize, and analogize. These are valuable acts. But the system must distinguish proposal from certificate. It must know when a transformation is heuristic, when it is undecided, when it is obstructed, and when it is formally admissible.

This distinction protects creativity. A purely restrictive system would refuse too much. A purely generative system asserts too much. An admissibility-aware system can explore freely while labeling its outputs. It can say: this is a conjectural analogy; this is a retrieved but unverified theorem; this is a formally checked proof; this is a valid substitution under group isomorphism; this is blocked by an invariant; this is true only after quotienting by π. That is the correct balance between creativity and rigor.

The slogan therefore works both for specialists and for the public. For specialists, it points to proof assistants, quotient formation, invariants, and typed certificates. For the public, it says that the next AI should not merely sound right. It should know what it is allowed to transform.

E.3 The Research Program in One Page

The research program of Admissibility Algebra for Mathematical AI can be summarized as follows.

First, define mathematical identity as policy-indexed. Replace unqualified sameness with the judgment $\Gamma \vdash X \equiv_\pi Y : C$. The context Γ defines jurisdiction. The objects X and Y are the things being compared. The policy π defines permitted sameness. The certificate C witnesses the admissibility. Negative cases are recorded as $\Gamma \nvdash X \equiv_\pi Y$ because $O_\pi(X,Y)$.

Second, treat policies as first-class mathematical objects. Equality, isomorphism, equivalence, homotopy equivalence, quotient equivalence, definitional equality, propositional equality, formalization fidelity, theorem applicability, and heuristic similarity are different policies. They have different certificates, invariants, obstructions, and substitution scopes. The system must not collapse them.

Third, build an admissibility database. The database stores MathematicalObject, Policy, Invariant, Certificate, Obstruction, Quotient, AdmissibilityJudgment, SourceDocument, FormalSystem, ProofTrace, ModelOutput, and VerificationStatus. Its atomic record is not a theorem alone, but a judgment about whether one object may stand in for another under a policy.

Fourth, connect the database to proof assistants. Lean, Coq, Agda, Isabelle, and related systems provide certificate checking. They do not replace the admissibility database, but they supply formal authority for many records. A proof assistant can promote an uncertified proposal into a verified certificate or generate failure traces that become obstructions.

Fifth, train mathematical AI on certified transformations. The model should not learn only $X \to Y$. It should learn Γ, X, Y, π, $I\pi$, C, O, and $S\pi$. It should see positive certificates, negative obstructions, undecided states, uncertified proposals, and contrastive near-misses. It should learn not merely what mathematicians say, but what mathematics permits.

Sixth, manage quotients explicitly. Every abstraction is a quotient-like act of forgetting. The system must know what was collapsed, what survived, whether operations descend, whether witnesses are retained, and what substitutions are licensed in Ω/π. This prevents destructive abstraction.

Seventh, export the discipline beyond mathematics only after it works in mathematics. Scientific reasoning, legal reasoning, memory systems, tool use, and planning all involve transformations under identity constraints. Mathematics is the clean seed domain because its certificates and obstructions can be made explicit.

The research program is therefore staged: theory, schema, prototype, proof-assistant integration, admissibility corpus, model training, quotient management, and generalization.

E.4 Minimal Public Explanation

A nontechnical reader can understand the idea through a simple example.

Suppose two mathematical objects both have four elements. An ordinary AI might say they are the same because they have the same size. But if the objects are groups, size is not enough. The cyclic group $\mathbb{Z}/4\mathbb{Z}$ and the group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ both have four elements, but they are not the same as groups. One has an element of order 4; the other does not. They are the same only under the

weak policy “same number of elements.” They are different under the stronger policy “same group structure.”

That is Admissibility Algebra in miniature. The question is not simply whether two things are the same. The question is: same under what policy, preserving what invariant, with what certificate, and for what use?

This example gives the public explanation:

Current AI often notices similarity.
Mathematical AI must know whether similarity permits replacement.
Replacement requires a policy.
A policy requires preserved invariants.
A valid replacement requires a certificate.
A failed replacement should produce an obstruction.
The answer should state the scope.

The public version should avoid overwhelming readers with formalism at first. The formal judgment $\Gamma \vdash X \equiv_\pi Y : C$ can be introduced after the intuition. It can be read as:

In context Γ, X and Y count as the same under policy π, with certificate C.

That sentence is sufficient for broad readers. It explains why AI needs more than embeddings, retrieval, and fluent generation.

E.5 Public Examples Beyond Algebra

Several public examples can make the program vivid.

First, theorem retrieval. A model finds a theorem that sounds relevant. But a relevant theorem is not necessarily applicable. Its hypotheses must match the current problem. If a compactness hypothesis is missing, the theorem cannot be used. The policy is theorem applicability; the invariant is hypothesis preservation; the obstruction is missing compactness.

Second, proof summarization. A model summarizes a long proof into a short paragraph. The summary may be readable, but does it preserve the proof’s logical skeleton? If the summary omits the only assumption that makes the argument true, the summary is a destructive quotient. The policy is proof-summary fidelity; the invariant is dependency preservation; the obstruction is omitted hypothesis.

Third, formalization. A model turns an informal theorem into Lean code. The Lean theorem may compile, but it might not say exactly what the informal theorem meant. Compilation certifies the formal theorem, not the translation. The policy is formalization fidelity; the invariants are

hypotheses, quantifiers, domain, definitions, and conclusion; the certificate is an alignment between informal and formal statements.

Fourth, analogy. A model notices that two arguments have the same shape. Analogy is useful for discovery, but it does not license substitution. The policy is heuristic similarity unless a stronger certificate is found. The correct output is not “therefore proved,” but “candidate analogy; certificate required.”

Fifth, quotienting. A model collapses many examples into a concept. That abstraction is useful only if it preserves the invariants needed later. If the collapse erases essential differences, reasoning fails. The policy is quotient formation; the certificate is well-definedness; the obstruction is loss of structure.

These examples show that Admissibility Algebra is not an exotic add-on. It describes the hidden discipline already required whenever AI handles serious knowledge.

E.6 Message for Mathematicians

For mathematicians, the message is that Admissibility Algebra makes explicit what expert practice already manages implicitly. Mathematicians constantly move among equality, isomorphism, equivalence, homotopy equivalence, quotienting, normal forms, presentations, and canonical representatives. They know, often tacitly, which properties survive which transformations. They know when “without loss of generality” is legitimate, when “up to isomorphism” is harmless, and when a quotient has destroyed too much information.

Mathematical AI does not possess this tacit discipline unless it is represented. Large models learn language patterns, but the permission structure is deeper than language. The admissibility database is a way to externalize that structure. It makes policies searchable, invariants explicit, certificates typed, obstructions reusable, and substitution scopes auditable.

The proposal is not to replace existing foundations. Set theory, category theory, type theory, homotopy type theory, and proof assistants remain essential. Admissibility Algebra sits above and across them as an identity-governance layer. It asks which identity policy is being used in a given transformation and what certificate licenses it. In that sense, it is not a rival to category theory or HoTT. It is a policy-indexing layer for their use in AI.

The message to mathematicians is also practical. Formal libraries already contain enormous admissibility data, but it is not indexed as such. Preservation theorems, rewrite rules, equivalences, quotient definitions, isomorphism structures, coercions, tactic traces, and failed proof attempts can become training data for policy-aware AI. The task is to re-index formal mathematics as admissible transformation.

A mathematician should be able to ask a future system:

Under what policy are these objects the same?
What invariant blocks stronger identity?
What certificate transports this theorem?
Which quotient did this abstraction form?
Where is substitution licensed?
Is this formalization faithful?
Is this proof step certified or only suggested?

These are mathematical questions. They should be first-class AI questions.

E.7 Message for AI Researchers

For AI researchers, the message is that mathematical hallucination is not merely a data-scale problem. It is an architecture problem. More data and larger models may improve fluency, but fluency does not by itself produce admissibility. The model needs a representation of permitted transformation.

A retrieval-augmented model may retrieve a relevant theorem, but retrieval does not prove applicability. A proof-generating model may produce a plausible tactic, but the tactic must be checked. A formalization model may produce compiling code, but compilation does not prove semantic fidelity to the informal source. A summarization model may produce a concise explanation, but concision does not prove invariant preservation.

The architecture proposed here separates proposal from certification. Neural models generate candidates. Vector databases retrieve related material. Graph databases expose policy and dependency structure. Relational databases store judgments. Proof assistants check formal certificates. Human experts annotate informal-formal fidelity where necessary. Obstruction records capture failure. This hybrid architecture is not a compromise; it is the natural structure of mathematical trust.

For training, the key change is to move beyond input-output pairs. Train on records containing context, objects, policy, invariants, certificates, obstructions, and scopes. Give the model contrastive examples where a transformation is valid under one policy and invalid under another. Reward refusal when no certificate exists. Reward obstruction diagnosis. Reward weakest-policy identification. Reward correct distinction between heuristic and certified status.

The AI research version of the slogan is:

Do not train only next-token mathematical fluency. Train policy-indexed admissibility.

This may allow smaller systems to become more trustworthy. A model with access to a well-structured admissibility database and proof checker may outperform a larger fluent model on reliability. The intelligence is not only in the weights. It is in the architecture of permission.

E.8 Message for Proof-Assistant Communities

For proof-assistant communities, the message is that their systems are already admissibility engines, but their libraries can be re-indexed for AI in a new way. Every theorem, proof term, rewrite rule, equivalence, quotient, coercion, typeclass instance, and tactic trace encodes lawful transformation. These are not only proofs. They are admissibility data.

A proof assistant already enforces local validity. But AI needs a higher-level index: which policy was used, which invariant was preserved, which substitution scope was licensed, and which obstruction occurred when a proof failed. Proof assistants can provide the trusted checking layer, but the admissibility database can provide the AI-facing memory layer.

This suggests practical projects. Extract preservation theorems from formal libraries. Index isomorphism and equivalence structures. Record rewrite rules with their required contexts. Preserve failed tactic traces as negative examples. Link quotient constructions to well-definedness certificates. Attach theorem applications to hypothesis-matching records. Build policy graphs connecting equality, isomorphism, equivalence, and quotient policies across domains.

The result would make formal libraries more useful not only for AI but for humans. A mathematician could navigate by admissibility: what does this equivalence preserve? Which theorem transports across it? Why did this rewrite fail? What stronger policy would be required? Which quotient has erased this distinction?

Proof assistants can become the certificate authorities of mathematical AI. The admissibility database can become the map of what those certificates permit.

E.9 Message for General AI Trust

For general AI trust, the message is that mathematics is the clean training ground for a broader principle. Many AI failures outside mathematics are also inadmissible transformations.

A system turns uncertainty into fact.
A system turns a user preference into a permanent memory.
A system turns a rough instruction into unauthorized action.
A system turns correlation into causation.
A system turns one jurisdiction’s law into another’s.
A system turns a retrieved source into a verified claim.

A system turns a stereotype into a category.
A system turns a summary into a replacement for evidence.

Each failure can be described as a transformation that exceeded its policy. The system did not preserve the invariant that mattered. It lacked a certificate. It formed a destructive quotient. It substituted outside scope.

Mathematics lets us build the discipline where the boundaries are clearest. Once the system can distinguish equality from isomorphism, isomorphism from equivalence, equivalence from analogy, proof from explanation, and certificate from plausibility, similar ideas can be exported to other domains. Scientific AI can index evidence policies. Legal AI can index jurisdiction and authority policies. Memory AI can index persistence and consent policies. Tool-using AI can index authorization policies.

The general principle is:

No transformation without a governing policy.
No assertion without appropriate certificate.
No abstraction without preserved invariants.
No substitution outside scope.

This is the broad trust thesis of Admissibility Algebra.

E.10 The Paper's Core Contributions

The paper contributes five core ideas.

First, it identifies admissibility as the missing layer in mathematical AI. Mathematical intelligence is not merely generating plausible expressions. It is knowing which transformations are permitted.

Second, it introduces the primitive judgment $\Gamma \vdash X \equiv_\pi Y : C$. This judgment makes context, policy, equivalence, and certificate explicit. It also supports negative and undecided variants.

Third, it treats hallucination as unauthorized substitution. This reframes AI error as policy failure, invariant loss, missing certificate, or destructive quotienting.

Fourth, it proposes an admissibility database. The database stores objects, policies, invariants, certificates, obstructions, quotients, judgments, sources, formal systems, proof traces, model outputs, and verification status.

Fifth, it proposes training mathematical AI on certified transformations. The model should learn X, Y, π, I_π, C, O, Γ, and S_π rather than only $X \to Y$.

These contributions form a coherent program. The theoretical layer defines admissibility. The database layer stores it. The proof-assistant layer checks it. The training layer learns it. The public slogan communicates it.

E.11 Short Abstract Version for Talks

A short talk abstract can be written as follows.

Current AI systems predict mathematical transformations but do not reliably certify whether those transformations are admissible. This paper proposes Admissibility Algebra as a policy-indexed foundation for mathematical AI. Its primitive judgment is $\Gamma \vdash X \equiv_\pi Y : C$, meaning that in context Γ, objects X and Y count as the same under policy π with certificate C. Policies include equality, isomorphism, equivalence, homotopy equivalence, quotient equivalence, formalization fidelity, theorem applicability, and heuristic similarity. Each policy specifies preserved invariants, certificate types, obstructions, and substitution scope. The framework treats hallucination as unauthorized substitution, abstraction as controlled quotienting, proof assistants as admissibility engines, and formal libraries as corpora of certified transformations. The proposed implementation is an admissibility database storing objects, policies, invariants, certificates, obstructions, quotients, and judgments. The goal is to train mathematical AI not merely on $X \to Y$, but on X, Y, π, I_π, C, O, Γ, and S_π. The future AI slogan is: current AI predicts transformations; mathematical AI must certify admissible transformations.

E.12 Short Public Version

A short public version can be written as follows.

Today's AI can often produce a plausible answer, proof sketch, summary, or analogy. But mathematics requires more than plausibility. It requires knowing when one thing may legitimately stand in for another. Two objects may be the same as sets but different as groups. Two spaces may be the same up to homotopy but different as topological spaces. Two theorem statements may sound alike but differ by a crucial hypothesis. Admissibility Algebra gives AI a way to manage these distinctions. It asks: same under what policy, preserving what invariant, with what certificate, and for what use? The core judgment is $\Gamma \vdash X \equiv_\pi Y : C$: in context Γ, X and Y count as the same under policy π with certificate C. This turns AI from a generator of plausible transformations into a checker of admissible transformations. The slogan is simple: current AI predicts transformations; future mathematical AI must certify them.

E.13 Closing Statement

Admissibility Algebra begins from a simple observation: mathematical reasoning is transformation under identity constraints. Every proof, quotient, rewrite, formalization, summary, theorem application, and abstraction depends on knowing what may be preserved

and what may be forgotten. Current AI systems transform fluently but often without authority. They predict what could come next, but they do not always know whether the move is licensed.

The remedy is not to suppress transformation. Transformation is intelligence. The remedy is to certify it. Policies define permitted sameness. Invariants define what must not change. Certificates witness lawful replacement. Obstructions explain failure. Quotients formalize controlled collapse. Proof assistants check formal cases. Databases store admissibility memory. Models trained on these records learn mathematical permission.

The final thesis is therefore:

Mathematical AI becomes trustworthy when it stops treating similarity as identity and begins treating identity as a certified, policy-indexed judgment.

Current AI predicts transformations. Mathematical AI must certify admissible transformations.